Lima Documentation

Release 1.3.1

Lima Team

December 15, 2015

CONTENTS

1 Installation **3**
- 1.1 Windows . . . 3
- 1.2 Linux . . . 9
- 1.3 Tango Device Server . . . 12

2 Howto contribute **19**
- 2.1 Prerequisite . . . 19
- 2.2 Contribute guideline . . . 20

3 Supported Cameras **23**
- 3.1 Windows Only . . . 23
- 3.2 Linux only . . . 35
- 3.3 Windows and Linux . . . 75

4 Future cameras **81**
- 4.1 Under development . . . 81
- 4.2 Foreseen . . . 81

5 Applications **83**
- 5.1 Python TANGO server . . . 83

6 Camera plugin developer guide **129**
- 6.1 Understand the plugin architecture . . . 129
- 6.2 Organize the source code . . . 134
- 6.3 Do not forget to write a documentation . . . 138

7 Class documentation **141**

8 Indices and tables **143**

Lima (**L** ibrary for **Im** age **A** cquisition) is a project for the unified control of 2D detectors. The aim is to clearly separate hardware specific code from common software configuration and features, like setting standard acquisition parameters (exposure time, external trigger), file saving and image processing.

A PDF version of this documentation is available: pdf documentation

Project's Specification is available *spec*

If you want to be in touch with LIMA community, please send email to lima@esrf.fr. You can also subscribe to the mail-list by sending a message to sympa@esrf.fr with as subject *subscribe lima*.

CHAPTER

ONE

INSTALLATION

1.1 Windows

1.1.1 Prerequisite

Before installing Lima on windows plateform, you need to install the following tools :

- Python 2.6 or more recent.
- gsl-1.8.
- Visual C++ 2008 Express.
- GitHub for windows.

1.1.2 GetIt

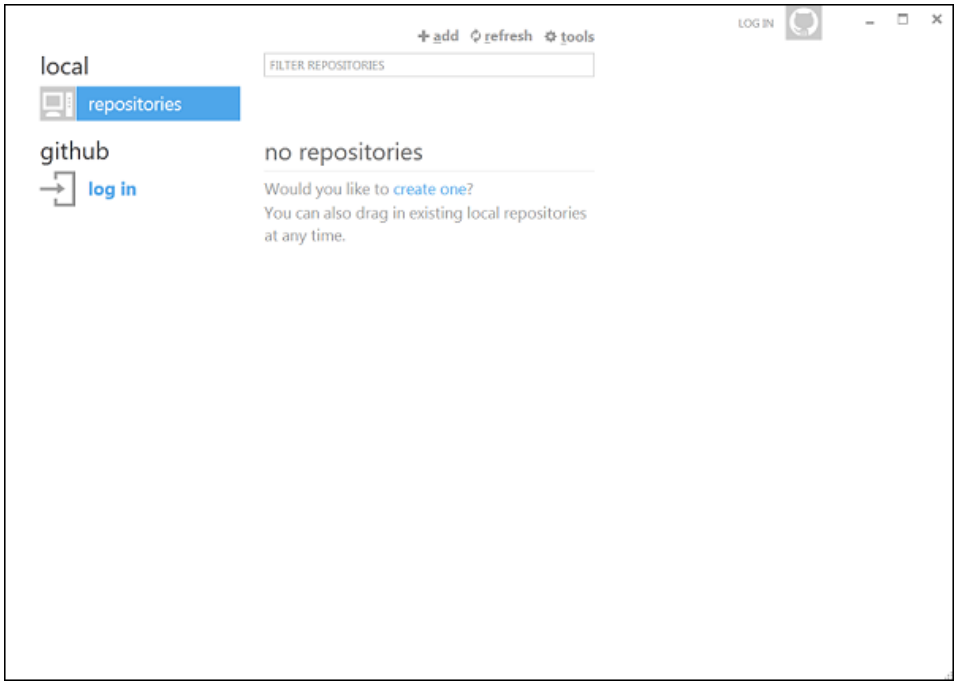
As Lima is not packaged,the only way for now is to retreived it from the git repository. Lima can be retreived using **GitHub** for Windows. There are two ways to get Lima :

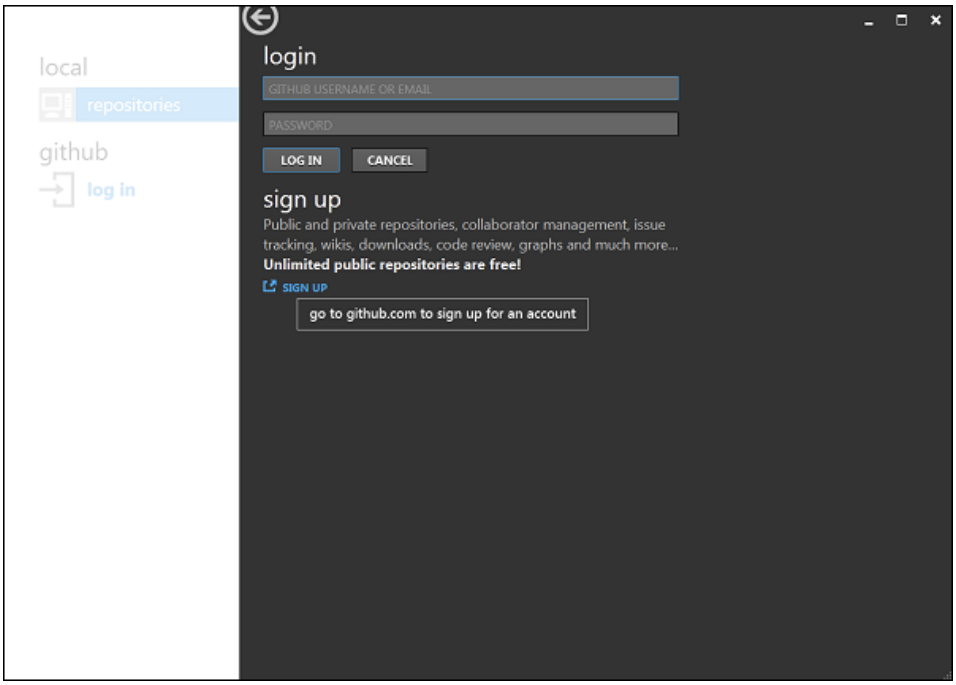
- For ordinary user, you can get the source code directly (zip file),
- For developpers who want to share their work with the community, you must create a fork and clone Lima.

The procedure to retreived is described below.

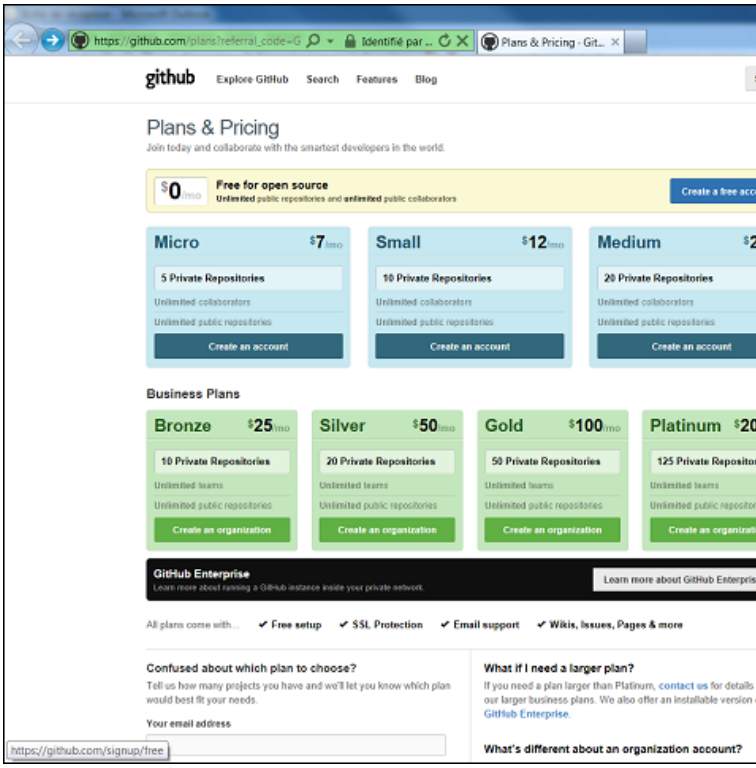
- **create an account**

  Launch GitHub form the desktop menu and click on "LOG IN".



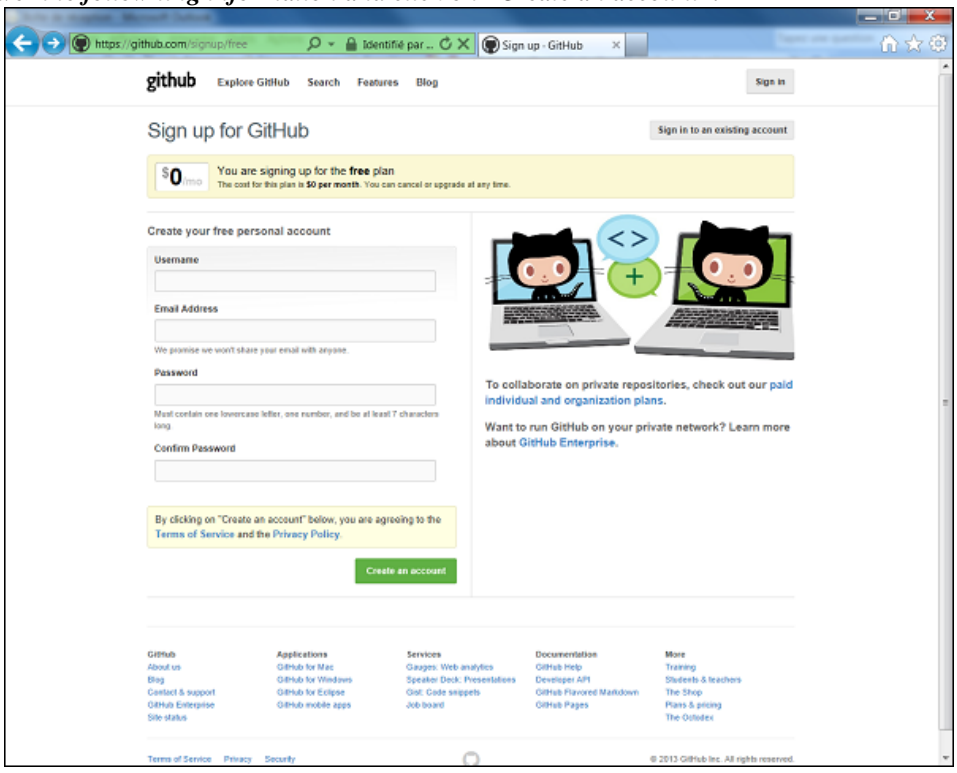


Click on "SIGN UP".



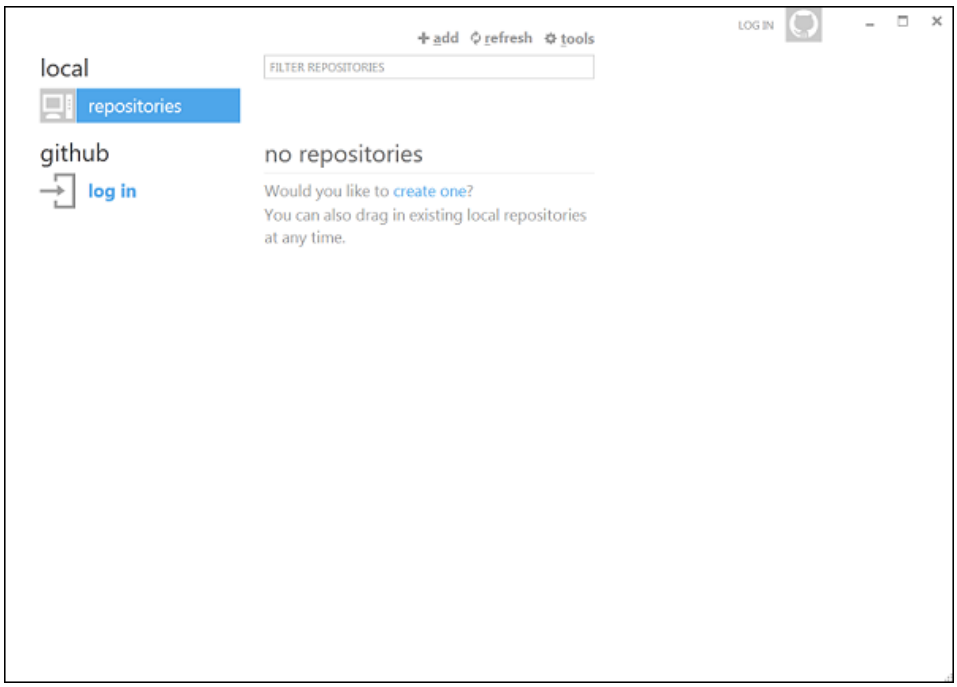
On the web page click on "Create a free account".

Answer the followning information and click on "Create an account".

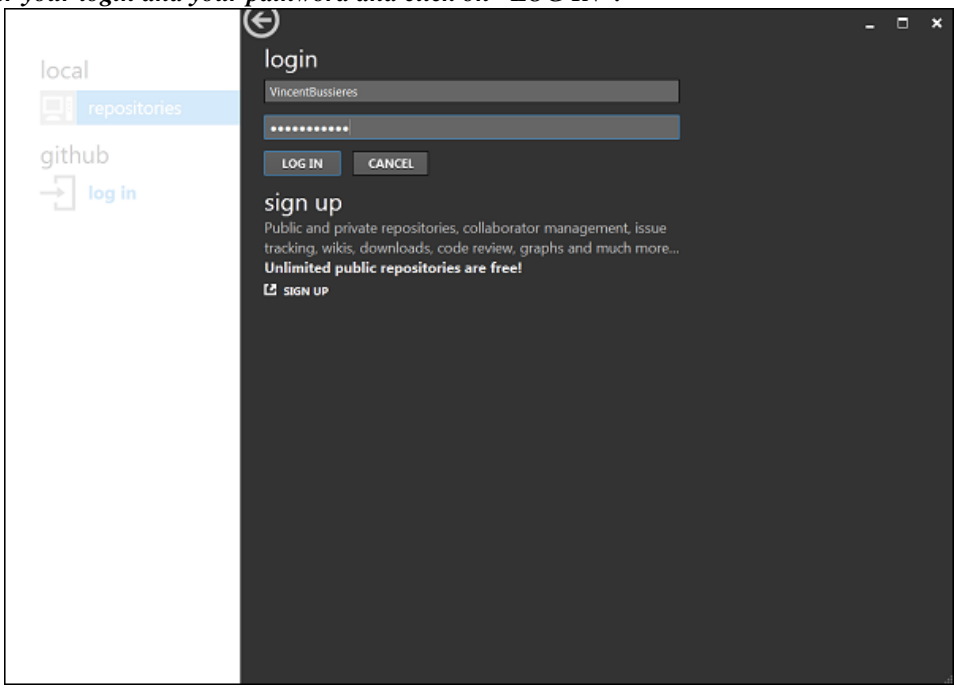

https://github.com/signup/free
Sign up · GitHub
github
Explore GitHub
Search
Features
Blog
Sign in
Sign up for GitHub
Sign in to an existing account
$0/mo
You are signing up for the free plan
Create your free personal account
Username
Email Address
Password
Confirm Password
By clicking on "Create an account" below, you are agreeing to the Terms of Service and the Privacy Policy.
Create an account
To collaborate on private repositories, check out our paid individual and organization plans.
Want to run GitHub on your private network? Learn more about GitHub Enterprise.

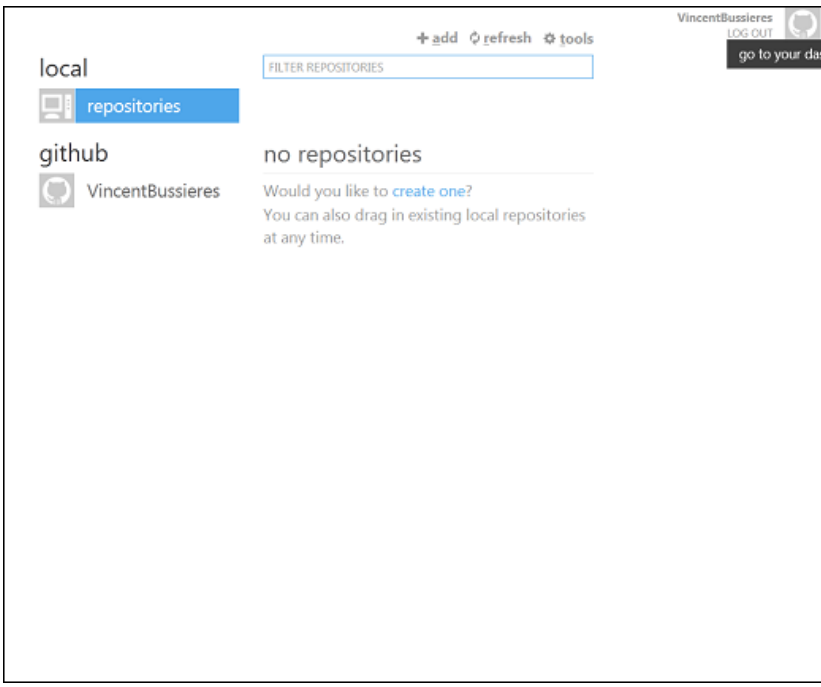
- **Get source code for ordinary user**

  Launch GitHub form the desktop menu and click on "LOG IN".



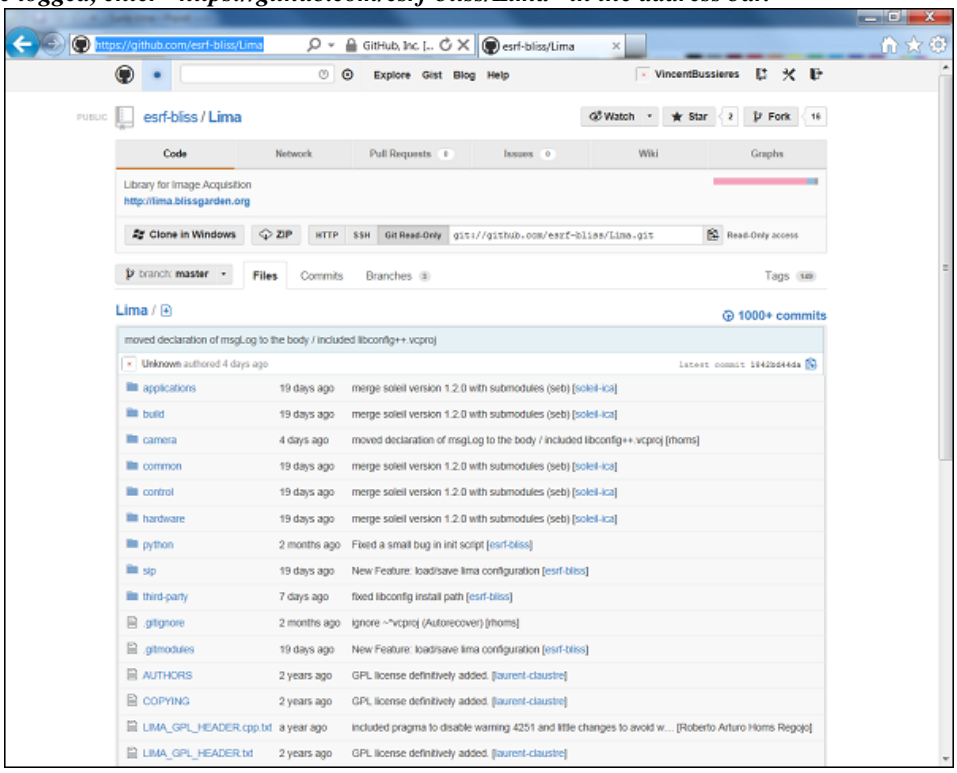
Enter your login and your pathword and click on "LOG IN".





Click on your login to go to your dashboard.

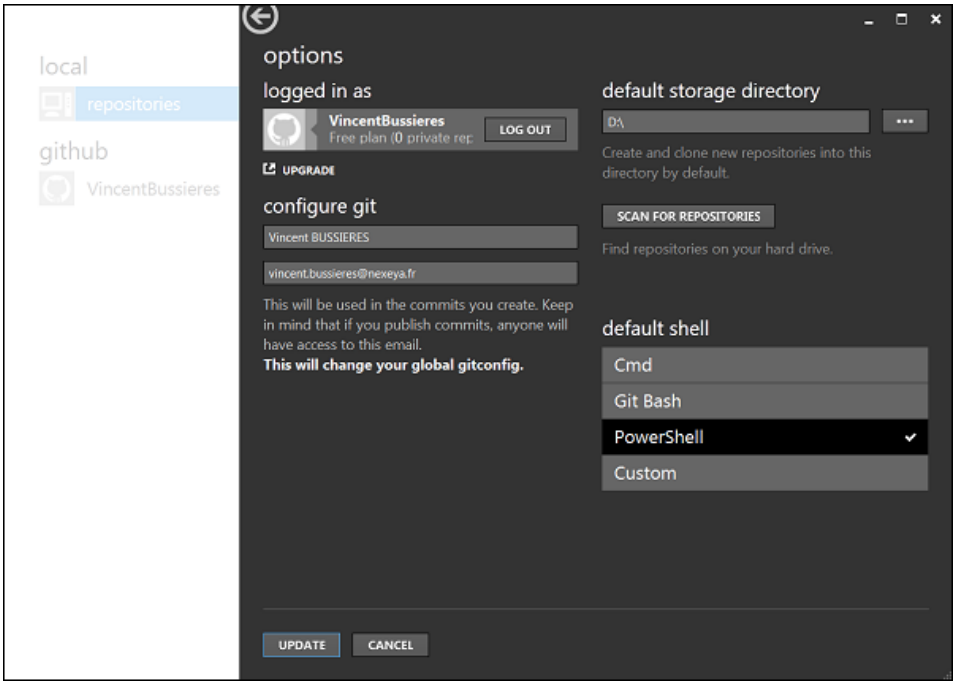
Once logged, enter "https://github.com/esrf-bliss/Lima" in the address bar.


https://github.com/esrf-bliss/Lima
esrf-bliss/Lima
Explore
Gist
Blog
Help
VincentBussieres
esrf-bliss / Lima
Watch
Star
Fork
Code
Network
Pull Requests
Issues
Wiki
Graphs
Library for Image Acquisition
http://lima.blissgarden.org
Clone in Windows
ZIP
HTTP
SSH
Git Read-Only
git://github.com/esrf-bliss/Lima.git
Read-Only access
branch: master
Files
Commits
Branches
Tags
Lima /
1000+ commits
moved declaration of msgLog to the body / included libconfig++.vcproj
Unknown authored 4 days ago
applications 19 days ago merge soleil version 1.2.0 with submodules (seb) [soleil-ica]
build 19 days ago merge soleil version 1.2.0 with submodules (seb) [soleil-ica]
camera 4 days ago moved declaration of msgLog to the body / included libconfig++.vcproj [rhoms]
common 19 days ago merge soleil version 1.2.0 with submodules (seb) [soleil-ica]
control 19 days ago merge soleil version 1.2.0 with submodules (seb) [soleil-ica]
hardware 19 days ago merge soleil version 1.2.0 with submodules (seb) [soleil-ica]
python 2 months ago Fixed a small bug in init script [esrf-bliss]
sip 19 days ago New Feature: load/save lima configuration [esrf-bliss]
third-party 7 days ago fixed libconfig install path [esrf-bliss]
.gitignore 2 months ago ignore ~*vcproj (Autorecover) [rhoms]
.gitmodules 19 days ago New Feature: load/save lima configuration [esrf-bliss]
AUTHORS 2 years ago GPL license definitively added. [laurent-claustre]
COPYING 2 years ago GPL license definitively added. [laurent-claustre]
LIMA_GPL_HEADER.cpp.txt a year ago included pragma to disable warning 4251 and little changes to avoid w... [Roberto Arturo Horns Regojo]
LIMA_GPL_HEADER.txt 2 years ago GPL license definitively added. [laurent-claustre]

Retrieved source code by clicking on "ZIP button".

- **Fork and clone Lima for developpers**

  Launch GitHub and log in. You can modify you default storage directory by selecting "Tools/Options..." menu.



Repeat the previous steps and create a fork by clicking on "Fork". You can now clone Lima repository on your default storage directory.

1.1.3 Compilation & Installation

Installation of Lima works on Windows XP and Windows 7.

Installation is done by following the steps below:

- Copy config.inc_default to config.inc in the Lima directory.
- Configure config.inc file for used cameras.

```
############################################################################
# This file is part of LImA, a Library for Image Acquisition
#
# Copyright (C) : 2009-2011
# European Synchrotron Radiation Facility
# BP 220, Grenoble 38043
# FRANCE
#
# This is free software; you can redistribute it and/or modify
# it under the terms of the GNU General Public License as published by
# the Free Software Foundation; either version 3 of the License, or
# (at your option) any later version.
#
# This software is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
# GNU General Public License for more details.
#
# You should have received a copy of the GNU General Public License
# along with this program; if not, see <http://www.gnu.org/licenses/>.
############################################################################

COMPILE_CORE=1
COMPILE_SIMULATOR=1
COMPILE_SPS_IMAGE=0
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=1
COMPILE_PROSILICA=0
COMPILE_ROPERSCIENTIFIC=0
COMPILE_MYTHEN=0
COMPILE_ADSC=0
COMPILE_UEYE=0
COMPILE_XPAD=0
COMPILE_PERKINELMER=0
COMPILE_ANDOR=0
COMPILE_PHOTONICSCIENCE=0
COMPILE_PCO=0
COMPILE_MARCCD=0
COMPILE_CBF_SAVING=0
COMPILE_NXS_SAVING=0
COMPILE_FITS_SAVING=0
COMPILE_EDFGZ_SAVING=0
COMPILE_CONFIG=1
LINK_STRICT_VERSION=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_BASLER COMPILE_PROSILICA COMPILE_ROPERSCIENTIFIC COMPILE_ADSC \
       COMPILE_MYTHEN COMPILE_UEYE COMPILE_XPAD COMPILE_PERKINELMER \
       COMPILE_ANDOR COMPILE_PHOTONICSCIENCE COMPILE_PCO COMPILE_MARCCD \
       COMPILE_CBF_SAVING COMPILE_NXS_SAVING COMPILE_FITS_SAVING \
```

- Run “install.bat” from the Lima directory. Installation takes a few minutes.

1.2 Linux

1.2.1 Prerequisite

Before installing Lima on Linux plateform, you need to install the following packages :

- Python 2.6 or more recent

- GCC
- Git
- Gnu Scientific Library, libgsl is available within most of the linux flavour
- sip tool (version >=4.2), used for generating the python binding

1.2.2 GetIt

As Lima is not packaged,the only way for now is to retreived it from the git repository.

Command to get all sources:

```
seb@pcbliss01:~/$ git clone --recursive git://github.com/esrf-bliss/Lima.git
```

Commands for a minimum chekout to get all source needed for a particular camera:

```
seb@pcbliss01:~/$ git clone git://github.com/esrf-bliss/Lima.git
seb@pcbliss01:~/$ cd Lima
seb@pcbliss01:~/Lima$ git submodule init third-party/Processlib third-party/Sps
seb@pcbliss01:~/Lima$ git submodule init camera/CAMERA_YOU_WANT_TO_COMPILE
seb@pcbliss01:~/Lima$ git submodule update
```

1.2.3 Particular version

On all version of LImA is tagged. So you can retreive a particular version.

This command provide the tag list:

```
seb@pcbliss01:~/Lima$ git tag -l
core-1.0.10
core-1.0.11
core-1.0.12
core-1.0.13
core-1.0.9
espia-1.0.4
espia-1.0.5
....
taco-camera-frelon-1.0.5
tango-camera-frelon-1.0.0
tango-camera-maxipix-1.0.2
tango-camera-maxipix-1.0.4
tango-camera-pilatus-1.0.2
tango-common-1.0.2
tango-common-1.0.3
tango-common-1.0.4
tango-common-1.0.5
tango-common-1.0.6
```

Let's say that you want the version 1.0.11 of Lima core with this command, you'll getit:

```
seb@pcbliss01:~/Lima$ git checkout core-1.0.11
seb@pcbliss01:~/Lima$ git submodule init
seb@pcbliss01:~/Lima$ git submodule update
```

1.2.4 Compilation

Every think is managed by root Makefile.

- So first generate the config.inc file.

```
make
```

- Edit the configuration file **config.inc**

```
COMPILE_CORE=1
COMPILE_SIMULATOR=1
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_CBF_SAVING
```

- Configure all module and stuff (don't need if you just want C++ libraries)

```
make config
```

- Finally compile all C++ libraries

```
make
```

- If you need Python modules

```
make -C sip -j3
```

That's all folks ;)

1.2.5 Installation

Installation on Linux is pretty easy because it's manage by Makefile. But those Makefile can only be used if you have compile everything including Python modules. Otherwise It'll failed. See *Compilation*

```
make install
```

you can specify the destination path with this variable **INSTALL_DIR**

With your new installation you may need to update your environment for both python and library paths:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<my-new-install-dir>/Lima/lib

export PYTHONPATH=$PYTHONPATH:<my-new-install-dir>
```

WARNING: *"make install"* only installed C++ libs and python modules, the application like the python Tango server (LimaCCDs) code remains under applications/tango. Please go to *Tango Device Server* for further instructions.

1.3 Tango Device Server

1.3.1 Server setup

As Tango server is provied as python script, you just have to copy the application/tango directory where you want.

- **camera directory:** contained all camera Tango device specifics so remove all none need script
- **doc directory:** contained plugins camera documentation (exhaustive list of properties, commands and attributes)
- **plugins directory:** contained all plugins device server like:
  - Roi counters
  - Mask...
- **scripts directory:** contained a script use at ESRF to start Lima device server (can also be removed)
- **LimaCCDs.py file:** python script to start Lima device server
- **LimaViewer.py file:** python script to start LimaViewer device server to get image from Lima device server

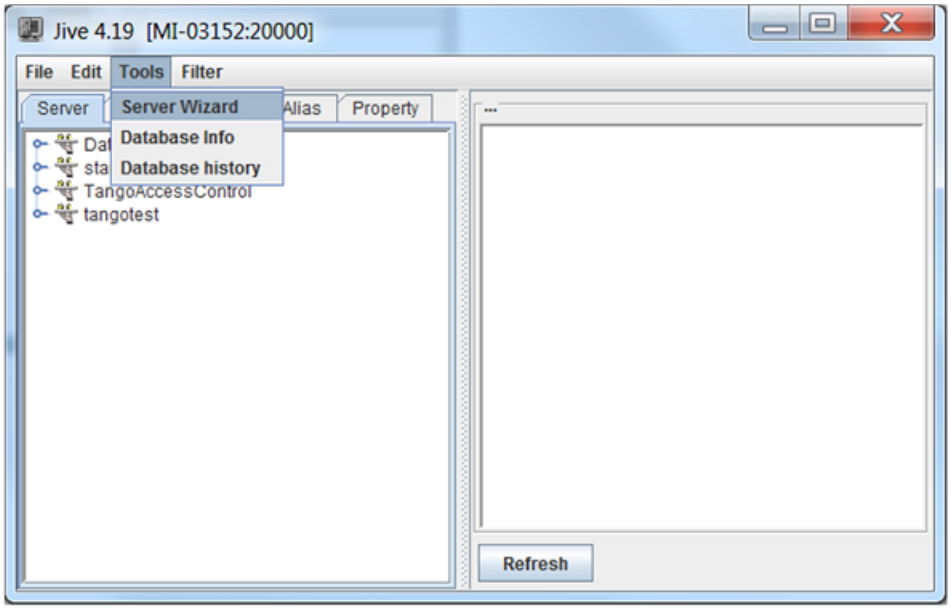
Warning: be sure your environment is properly set for python and library paths, see *Installation* for more information.

1.3.2 Example of plugin server setup : Basler detector

This procedure described the way to implement basler camera plugin. It is the same for whole the plugins, only properties may change.

You need to create a device server for Lima and another for the camera plugin. Lima device will use basler device thanks to "LimaCameraType" property. This property corresponds to the name of the camera plugin.

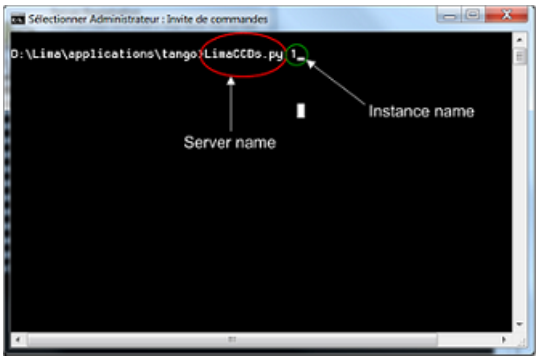
- **Lima device server:**
  - Run Jive and select "Tools->Server Wizard" menu. You must enter server and instance names



Tango Device Installation Wizard on MI-03152:20000
Server Registration
This wizard helps you to install a Tango device. First, you have to enter the "Server name" (executable file name) and its "instance name".
To register the server, click [Next].
Executable name
Server name
LimaCCDs
Instance name
1
< Back
Next >
Cancel

Click Next...

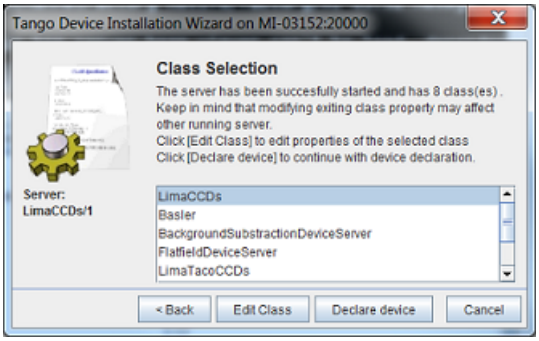
– Start the Lima device server. Open a terminal and execute the command “server_name instance_name”



Click Next on the “Tango Device Installation Wizard” window

– Declare a Lima device

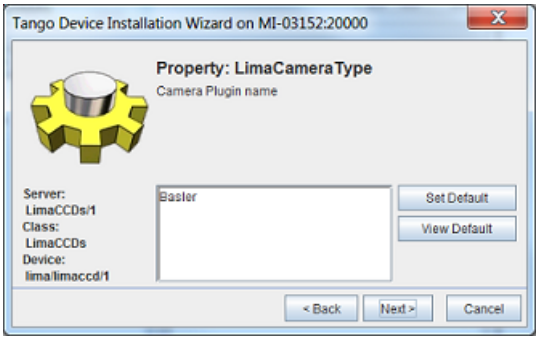
The Lima device server, contained several classes. For Basler camera you need to configure LimaCCDs and Basler classes.



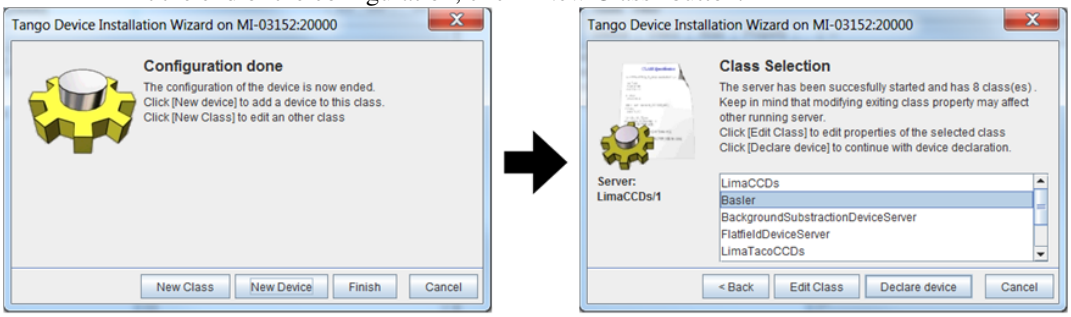
Select “LimaCCDs” class and click “Declare device” button. You must enter the device name with a string as “Domain/Family/member”.

Tango Device Installation Wizard on MI-03152:20000
Device Declaration of Class "LimaCCDs"
It is now time to give a device name for the 'LimaCCDs' class. If this device does not already exit, it will be added. Otherwise, the server's device list remains unchanged.
Click [Next] to edit device properties.
Device name
lima/limaccd/1
< Back
Next >
Cancel

Click Next and configure all the properties. You can let the default property values except for “LimaCameraType”. This property must contain the name of the Camera Plugin “Basler”.



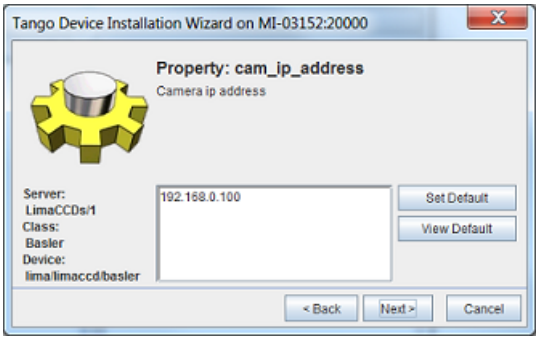
At the end of the configuration, click “New Class” button.


Tango Device Installation Wizard on MI-03152:20000
Configuration done
The configuration of the device is now ended.
Click [New device] to add a device to this class.
Click [New Class] to edit an other class
New Class
New Device
Finish
Cancel
Tango Device Installation Wizard on MI-03152:20000
Class Selection
The server has been succesfully started and has 8 class(es).
Keep in mind that modifying exiting class property may affect other running server.
Click [Edit Class] to edit properties of the selected class
Click [Declare device] to continue with device declaration.
Server:
LimaCCDs/1
LimaCCDs
Basler
BackgroundSubstractionDeviceServer
FlatfieldDeviceServer
LimaTacoCCDs
< Back
Edit Class
Declare device
Cancel

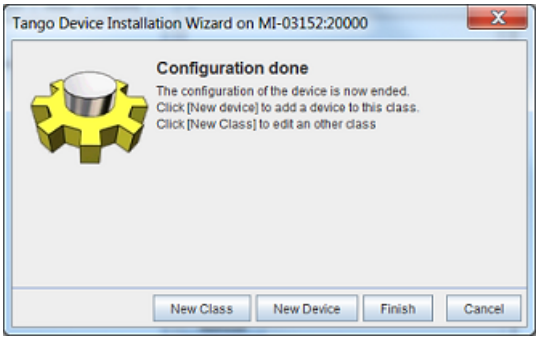
Select “Basler” class and click “Declare device” button. You must enter the device name with a string as “Domain/Family/member”.

Tango Device Installation Wizard on MI-03152:20000
Device Declaration of Class "Basler"
It is now time to give a device name for the 'Basler' class. If this device does not already exit, it will be added. Otherwise, the server's device list remains unchanged.
Click [Next] to edit device properties.
Device name
lima/limaccd/basler
< Back
Next >
Cancel

Click Next and configure all the properties. You can let the default property values except for "cam_ip_adress". This property must contain the IP adress of the Basler camera.



Configuration is now ended, click "Finish"


Tango Device Installation Wizard on MI-03152:20000
Configuration done
The configuration of the device is now ended.
Click [New device] to add a device to this class.
Click [New Class] to edit an other class
New Class
New Device
Finish
Cancel

- **Lima Viewer:**

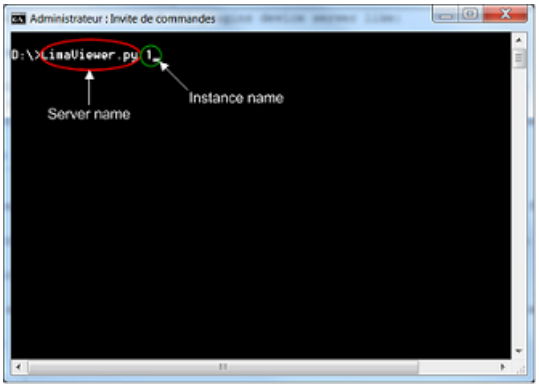
  To test the Lima device server, you can use the LimaViewer. This is a device server which periodically get the last image from the buffer. It allows the user to check that Lima device server is operational. The procedure below describe how to install and configure the LimaViewer device server.

  – Run Jive and select "Tools->Server Wizard" menu. You must enter server and instance names

Tango Device Installation Wizard on MI-03152:20000
Server Registration
This wizard helps you to install a Tango device. First, you have to enter the "Server name" (executable file name) and its "instance name".
To register the server, click [Next].
Executable name
Server name
LimaViewer
Instance name
1
< Back
Next >
Cancel

Click Next...

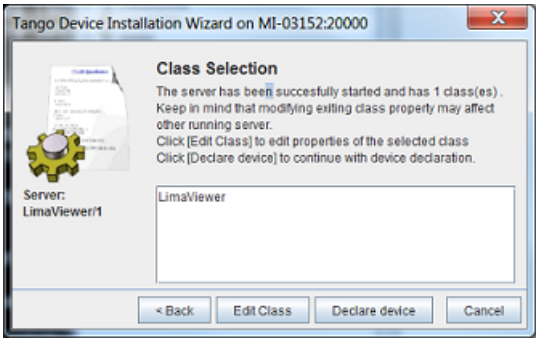
– Start the LimaViewer device server. Open a terminal and execute the command “server_name instance_name”



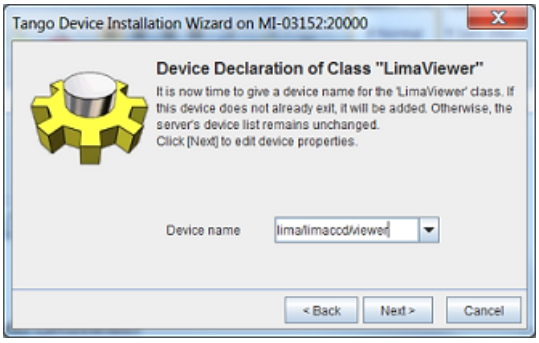
Click Next on the “Tango Device Installation Wizard” window

– Declare a LimaViewer device

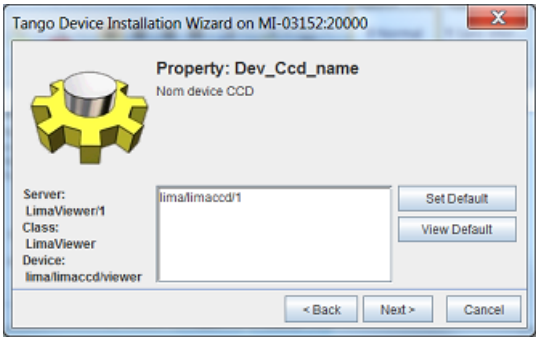
Select “LimaViewer” class and click “Declare device” button.


Tango Device Installation Wizard on MI-03152:20000
Class Selection
The server has been succesfully started and has 1 class(es).
Keep in mind that modifying exiting class property may affect other running server.
Click [Edit Class] to edit properties of the selected class
Click [Declare device] to continue with device declaration.
Server:
LimaViewer/1
LimaViewer
< Back
Edit Class
Declare device
Cancel

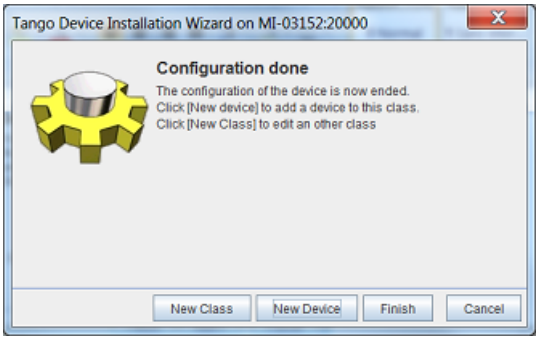
Enter the device name with a string as “Domain/Family/Member”.


Tango Device Installation Wizard on MI-03152:20000
Device Declaration of Class "LimaViewer"
It is now time to give a device name for the 'LimaViewer' class. If this device does not already exit, it will be added. Otherwise, the server's device list remains unchanged.
Click [Next] to edit device properties.
Device name
lima/limaccd/viewer
< Back
Next >
Cancel

Click Next and configure the “Dev_Ccd_name” property. This property corresponds to the name of the Lima device created before.

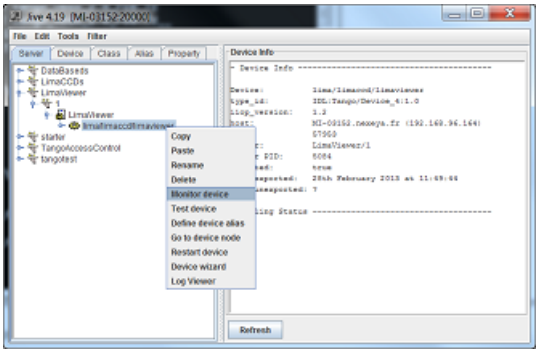


Configuration is now finished, click on "Finish"

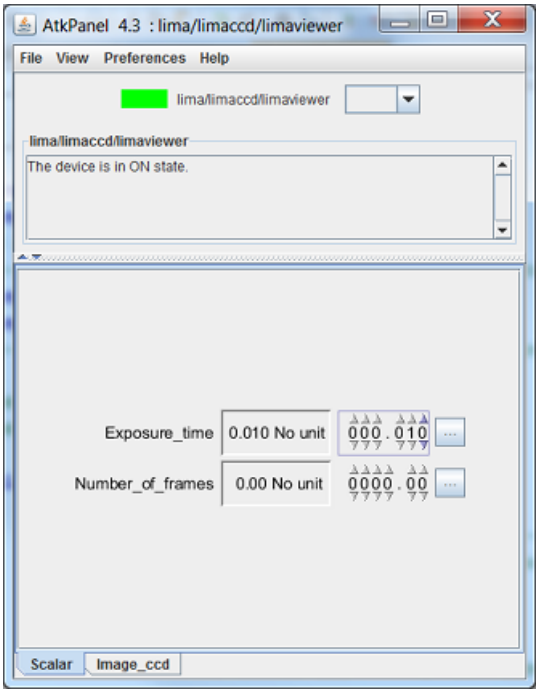

Tango Device Installation Wizard on MI-03152:20000
Configuration done
The configuration of the device is now ended.
Click [New device] to add a device to this class.
Click [New Class] to edit an other class
New Class
New Device
Finish
Cancel

- **Test LimaCCDs device server with Jive:**

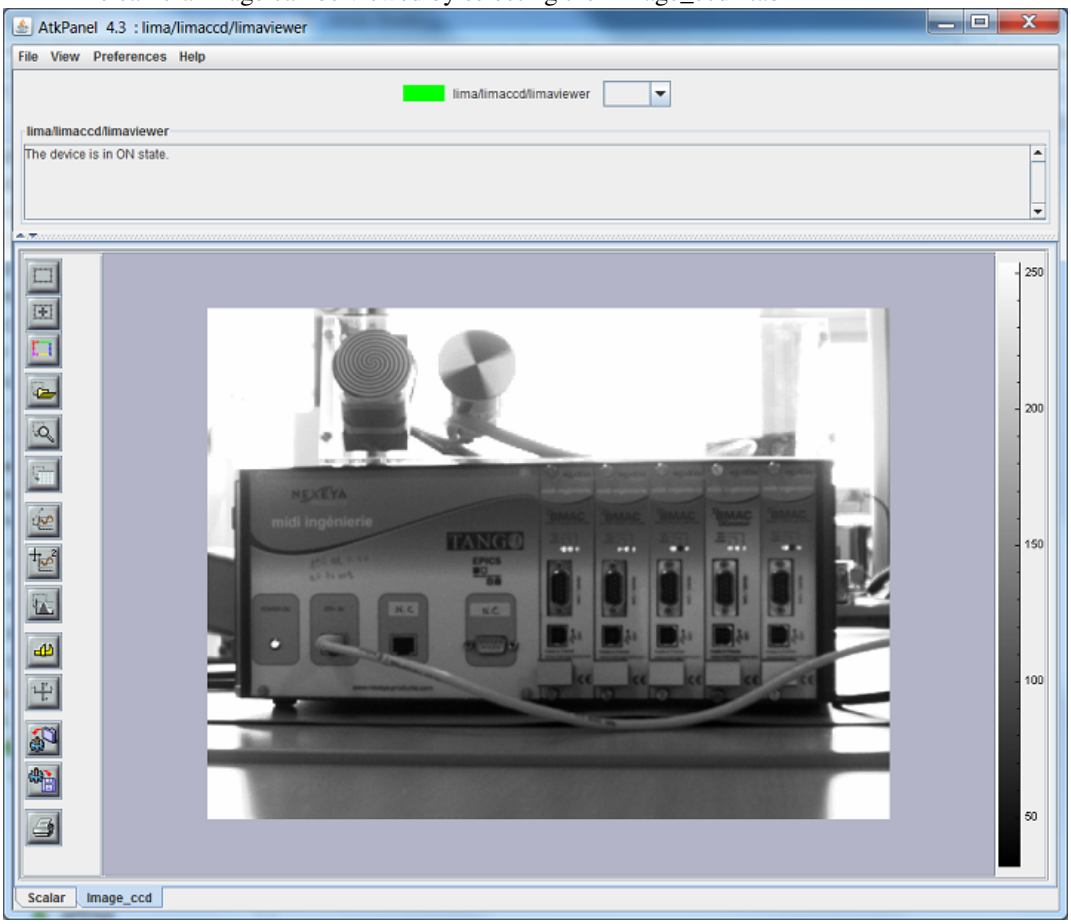
  The LimaViewer device appears in the Device tab from Jive. Make a right click on the LimaViewer device server and select "Monitor Device"



AtkPanel is now launched. You can configure exposure time and the number of frames to acquire.



The camera image can be viewed by selecting the “image_ccd” tab



CHAPTER

TWO

HOWTO CONTRIBUTE

2.1 Prerequisite

For collaborative development, we use the "Fork & Pull" model from github.com. So anyone who wants to contribute needs an account on github.com. Then you need to fork the project you want to contribute.

If you want to contribute with a new camera plug-in you should first request us (by email @ lima@esrf.fr) to get the new
plug-in camera sub-module created. We will provide:

- a default structure of directories (<mycamera>/src /include sip/ doc/ python/ test/)
- the Makefile files (<mycamera>/Makefile, <mycamera>/src/Makefile, <mycamera>/test/Makefile)
- templates files (src and include) for the mandatory classes:
- <MyCamera>Interface
- <MyCamera>DetInfoCtrlObj
- <MyCamera>SyncCtrlObj
- a standard .gitignore file
- a template index.rst for the documentation

In addition we will update with new entries other files needed to compile the plugin library and the python module:

- Lima/config.inc_default
- Lima/camera/Makefile
- build/Makefile
- sip/Makefile
- sip/configure.py

  As above do not forget to fork the new sub-module project.

2.1.1 Create a github account

This is an easy task, you have to go to Sign up - Github

2.1.2 Fork a project

On the github site, It is far better explained than we could do ;)

2.2 Contribute guideline

It is very simple to contribute, you should follow the steps below.

1. Branch

   First of all you have to create a branch for a new feature or for a bug fix, use an explicit branch name, for instance "soleil_video_patch" .

2. Code/patch

   If it's a patch from an existing module, respect and keep the coding style of the previous programmer (indentation,variable naming,end-line...).
   If you're starting a new camera project, you've just to respect few rules:

   - Class member must start with '**m_**'
   - Class method must be in **CamelCase**
   - You must define the camera's namespace

3. Commit

   Do as many commit as you need with clear comments.
   Prefer an atomic commit with a single change rather than a huge commit with too many changes.

4. Pull Request

   Then submit a pull request

   At this stage you have to wait, we need sometimes to accept or reject your request.
   So there are two possible issues:

   1. The Pull-request is accepted:

      we merge your branch with the the main project master branch, then everything is fine and you can now synchronize your forked project with the main project.

   2. The pull-request is rejected:

      The pull request could be rejected if:

      - the new code doesn't compile
      - it breaks backward compatibility
      - the python wrapping is missing or not updated

- the commit log message doesn't describe what you actually do

In case of a new camera plug-in sub-module the first pull request will be rejected if:

- as above
- the documentation is missing or if it does not fit with the guidelines (e.i *Understand the plugin architecture*)

We will tell you (github interface and/or email) about the reason and we will give some advises to improve your next tentative of pull-request.

So at this point you have to loop to item 2 (Code/Patch) again.
Good luck !!

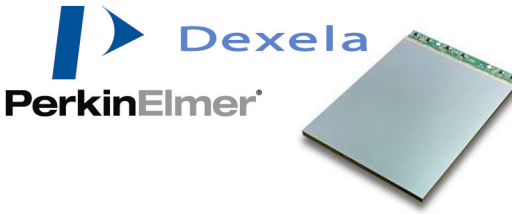
CHAPTER

THREE

SUPPORTED CAMERAS

3.1 Windows Only

3.1.1 Dexela camera plugin



Introduction

The Dexela detector is a brand product of PerkinElmer. PerkinElmer has recently Acquired Dexela Limited a manufacturer of CMOS flat panel. Nevertheless the Dexela detector SDK still remains not compatible with the other PerkinElmer detector SDK (see perkinelemer plugin) and one need to use this camera plugin instead.

Prerequisite

The Dexela detector model sensor2923 has only been tested at ESRF and it only works under **Windows 7 64 bits**.

The software Suite called XCAP Imaging version 3.0 has been installed on the computer. XCAP provides the SDK SDK-1.0.8.10-20.0.14.

The detector is controlled via an acquisition board: PIXCI(R) E4 PCIExpress Camera Link board (EPIX,Inc.).

The detector is now used in operation on the Swiss-Norvegian beamline BM01.

Installation & Module configuration

- follow first the steps for the windows installation *GetIt*

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=0
COMPILE_PROSILICA=0
COMPILE_ROPERSCIENTIFIC=0
COMPILE_MYTHEN=0
COMPILE_ADSC=0
COMPILE_UEYE=0
COMPILE_XH=0
COMPILE_XSPRESS3=0
COMPILE_XPAD=0
COMPILE_PERKINELMER=0
COMPILE_ANDOR=0
COMPILE_PHOTONICSCIENCE=0
COMPILE_PCO=0
COMPILE_MARCCD=0
COMPILE_POINTGREY=0
COMPILE_IMXPAD=0
COMPILE_DEXELA=1
COMPILE_RAYONIXHS=0
COMPILE_CBF_SAVING=0
COMPILE_NXS_SAVING=0
COMPILE_FITS_SAVING=0
COMPILE_EDFGZ_SAVING=0
COMPILE_TIFF_SAVING=0
COMPILE_CONFIG=1
LINK_STRICT_VERSION=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
      COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
      COMPILE_BASLER COMPILE_PROSILICA COMPILE_ROPERSCIENTIFIC COMPILE_ADSC \
      COMPILE_MYTHEN COMPILE_UEYE COMPILE_XH COMPILE_XSPRESS3 COMPILE_XPAD COMPILE_PERKINELMER \
      COMPILE_ANDOR COMPILE_PHOTONICSCIENCE COMPILE_PCO COMPILE_MARCCD COMPILE_DEXELA\
      COMPILE_POINTGREY COMPILE_IMXPAD COMPILE_RAYONIXHS COMPILE_CBF_SAVING COMPILE_NXS_SAVING \
      COMPILE_FITS_SAVING COMPILE_EDFGZ_SAVING COMPILE_TIFF_SAVING COMPILE_CONFIG\
      LINK_STRICT_VERSION
```

- start the windows compilation *Compilation & Installation*
- finally with the Tango server installation *Tango Device Server*

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

The camera will be initialized within the **DexelaInterface** object. There are 2 parameters to pass to the DexelaInterface() constructor:

- database_path (e.g. "C:\DexelaConfig.cfg")

- sensor_format (e.g. “sensor2923”)

Std capabilites

This plugin has been implemented in respect of the mandatory capabilites but with limitations according due to the detector specific features and with some programmer’s choices. We do not explain here the standard Lima capabilites but you can find in this section the useful information on the Dexela specfic features.

- HwDetInfo

  The Dexela detector as a pixel size of 74.8e-6 m (74.8 um) and the image data type is fixed to 16bpp (bit per pixel).

- HwSync

  The supported trigger modes are IntTrig, IntTrigMult, ExtTrigMult and ExtGate.

  The exposure time range is 0.0116 (1/86) to 120 seconds.

  The latency time is not manage.

Optional capabilites

In addition to the standard capabilities, we make the choice to implement some optional capabilities in order to have an improved simulation.

- HwShutter

  There is no shutter capability.

- HwRoi

  There is no hardware capability, but Lima provides the sofware Roi as well.

- HwBin

  The supported hardware binning factors are 1x1, 1x2, 1x4, 2x1, 2x2, 2x4, 4x1, 4x2 and 4x4.

Configuration

The main configuration will consist in providing the correct “DexelaConfig.cfg” file to the detector API. The file has to be provided by the manufacturer with a second file like “sensor2923.fmt”. The .fmt file contains some calibration data.

How to use

The LimaCCDs tango server provides a complete interface to the dexela plugin so feel free to test.

For a quick test one can use python, is this a short code example:

```
from Lima import Dexela
from lima impor Core
import time

hwint = Dexela.Interface('c:\DexelaConfig.cfg','sensor2923')
control = Core.CtControl(hwint)

acq = control.acquisition()
```

```
# setting new file parameters and autosaving mode
saving=control.saving()

pars=saving.getParameters()
pars.directory='/tmp/'
pars.prefix='testdexela_'
pars.suffix='.edf'
pars.fileFormat=Core.CtSaving.EDF
pars.savingMode=Core.CtSaving.AutoFrame
saving.setParameters(pars)


# now ask for 2 sec. exposure and 10 frames
acq.setAcqExpoTime(2)
acq.setNbImages(10)

acq.prepareAcq()
acq.startAcq()

# wait for last image (#9) ready
lastimg = control.getStatus().ImageCounters.LastImageReady
while lastimg !=9:
  time.sleep(1)
  lastimg = control.getStatus().ImageCounters.LastImageReady

# read a image
im0 = control.ReadImage(0)
```

3.1.2 Perkin Elmer

Intoduction

PerkinElmer is a world leader in the design, development, and manufacture of Amorphous Silicon (aSi) Flat Panel Detectors (FPD) designed to perform across a wide range of medical, veterinary, and industrial, Non-Destructive Testing (NDT) applications. Our XRD family of detectors provide superior image resolution, high frame rates up to 30 frames per seconds (fps), energy levels form 20 keV -15 MeV and easy information storage and retrieval.

The detector model we tested (ESRF) is : XRD 1621 CN ES

Module configuration

Previously to this you have to install the Perkinelmer SDK to the default path

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_PERKINELMER=1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_PERKINELMER COMPILE_CBF_SAVING
```

See *Installation*

Installation

- After installing perkinelmer modules *Installation*
- And probably Tango server *Tango Device Server*

3.1.3 PCO

Intoduction

- **PCO camera systems**
- PCO develops specialized fast and sensitive video camera systems, mainly for scientific applications; which covers digital camera systems with high dynamic range, high resolution, high speed and low noise. PCO home page
- **Product overview and technical data of the PCO cameras supported in LIMA**
- **PCO.dimax:** High speed 12 bit CMOS camera with fast image rates of 1469 frames per second (fps) at full resolution of 1920 x 1080 pixel. (tech data)
- **PCO.edge:** Extremely low noise sCMOS camera with fast frame rates (100 fps), wide dynamic range (1:27000), high quantum efficiency, high resolution (2560 x 2160) and large field of view. (tech data)
- **PCO.2000:** High resolution (2048 x 2048 pixel) and low noise 14bit CCD cooled camera system with internal image memory (camRAM), allows fast image recording with 160 MB/s. The available exposure times range from 500 ns to 49 days. (tech data)
- **PCO.4000:** High resolution (4008 x 2672 pixel) and low noise 14bit CCD cooled camera system with internal image memory (camRAM), allows fast image recording with 128 MB/s. The available exposure times range from 5 us to 49 days. (tech data)
- **Interface buses**
- **Cameralink:** used by **PCO.dimax** and **PCO.edge**
- **GigE:** used by **PCO.2000** and **PCO.4000**
- **Type of applications**
- Mainly used in scientific applications.
- **OS supported**

- **Win7 Professional** (english) 32 bits SP1.

Prerequisites

- **Required software packages**
  - **download links**
  - PCO and Silicon Software download (login/pw required)
  - VC++ download
  - GSL download
  - python download
  - numpy download
  - PyQt download
  - PyTango download
  - GIT download
  - **md5 checksum and size of packges used (maybe not updated)**

```
Silicon Software Runtime 5.1.4
    545a30a4aa07260072615257c5c983cd   58881018 DD_DCCLSISORT514WIN32_103.zip
    ca22c0385fc63b92e259d28b11b65021   58930007 RuntimeSetup_v5.1.4_IA32.exe

pco-sdk 1.15
    ===> select: silicon softwre section dll meIV
    89619de780943569dcfc885cb0a720d8   11608288 SW_PCOSDKWIN_115.exe
    bb030a743efd0954bb60b05dbfe3c45e   11560659 SW_PCOSDKWIN_115.zip

camware 3.08
   ===> select: silicon softwre section dll meIV
   8bd64a957c57287c4c30c0ce7811343e     12037008 SW_CAMWAREWIN32_308.exe
   ecf7936f451ed3a17cabbff7aa4ec645     11993859 SW_CAMWAREWIN32_308.zip

VC++ express edition 2008 with SP1
    62f764849e8fcdf8bfbc342685641304     2728440 vcsetup.exe

GSL GNU Scientific Lib (required for Process Lib)
    d0c114f842933622a156bfa757ee14c2     6476814 gsl-1.8.exe

Python / SIP / modules
    80b1ef074a3b86f34a2e6b454a05c8eb    15227904 python-2.6.6.msi
    267627e760277e5f6a74a83772a4f1d0     2342045 numpy-1.5.1.win32-py2.6-nosse.exe
    97a9c9b88fac273e1dfb4de9d0dce4ea    27712518 PyQt-Py2.6-x86-gpl-4.8.6-1.exe
    dc5200e3199947a9574014537905a4b7     3670016 PyTango-7.1.1.win32-py2.6.msi

GIT
    a8ab298fb7a728d41f7d787aef7dc8d1    14026948 Git-1.7.7.1-preview20111027.exe
```

- **Enviroment variables**
- **system variables**

```
===> add manually the python path (it is not set by the installation program)
     PATH -> C:\Python26;

===> used for some utility batch files
     PATH -> C:\blissadm\bat;
```

- **user variables**

```
TANGO_HOST -> <host>:20000
```

Module configuration

Configuration file **Lima/config.inc**

```
===> set these values to 1
     COMPILE_CORE=1
     COMPILE_PCO=1
```

See *Installation*

Post installation actions

- **enable/disable PCO logs**

```
===> rename file extensions (C:\ProgramData\pco):
     .txt (disabled) / .log (enabled) ----+
                                  camware.log   <---- created by hand
                                 PCO_CDlg.log
                                 PCO_Conv.log
                                  SC2_Cam.log
```

- **Command prompt console (Visual Studio)**

```
> All Programs
  > Microsoft Visual C++ 2008 Express Edition
    > Visual Studio Tools
      > Visual Studio 2008 Command Prompt
```

- **TODO**
- After installing PCO modules *Installation*
- And probably Tango server *Tango Device Server*

Configuration

- **TODO**

PCO EDGE notes

PC characteristics (used for PCO EDGE at ESRF)

- **PROCESSOR**

```
2x Intel Xeon E5645 Six-Core CPU, 2,40GHz, 80W, Socket LGA1366, 12MB 5,86GT/sec

CPU's: 2x Xeon SixCore E5645 2,40Ghz 12MB 5,86GT/sec
Intel Xeon E5645 Six-Core CPU, 2,40GHz, 80W, Socket LGA1366, 12MB
external cache. 5,86GT/sec QPI speed. 1333MHz memory speed (DDR3 only).
Intel Technologies: Intel Turbo Boost , Intel Hyper-Threading
Technology, Intel Virtualization (VT-x), Intel Trusted Execution,
Enhanced Intel SpeedStep, Intel Demand Based Switching, Execute
Disable Bit.
```

- **RAM**

```
24 GB (6x DDR3-1333 Reg. ECC 4 GB module)
```

- **HD**

```
C:
  WDC WD5003ABYX-01WERA1
  Western Digital 500 GB, 7200 RPM, SATA 2, 300 Mbps

D:
  Adaptec RAID 5405/5405Q with 2 HD of 450 Gb -> RAID0 837 GB
  HUS156045VLS600
  Hitachi 450GB, 15,000RPM SAS / Serial Attached SCSI, 6Gbps
```

- **graphic card**

```
Matrox G200eW
```

- **motherboard**

```
Motherboard Extended ATX format 13,68in x 13in, (34,7cm x 33cm) (W x H);
2 socket LGA 1366-pin. It supports processors Quad-Core Intel Xeon
series 5500; QPI bus system (up to 6.4GT/s); *chipset Intel 5520*;

18 socket DIMM 240 pin, support for up to 288GB memory DDR3
1333/1066/800MHz Registered or 48GB memory DDR3 unbuffered ECC, the real
operating ram speed depends on the processor?s model and number of
installed ram, best performances are achieved through a triple channel
configuration;
```

- **PCI slots**

```
1x PCIe x4 (in x8 slot)
3x PCIe x8
1x PCIe x8 (in x16 slot)
2x PCIe x16
```

PCO EDGE - install instructions for Silicon Software Me4 Board

Check the document **camera/pco/doc/Me4_Installation_Test_e1.pdf** with the requirements and procedure to install the CameraLink grabber card. It is important in order to get the maximum transfer speed required by the PCO EDGE camera.

The boards tested by PCO are:

```
Supermicro X8ST3
GigaByte GA-X58A-UD3R
Intel S5520
Intel DX58SO2
Supermicro X8DTH-iF
```

With the PC described in *PCO EDGE notes* the speed of the CameraLink is about **570 MB/s** (66% of the theoretic max of 860 MB/s).

PCO EDGE - shutter mode (global/rolling)

```
cam.talk("rollingShutter 0")   <--- set shutter mode to GLOBAL

cam.talk("rollingShutter 1")   <--- set shutter mode to ROLLING
```

After the change of the shutter mode, the cam is rebooted and requires about 10s to became ready, meanwhile the acq status is AcqConfig.

The validRanges (exposure and latency time) are updated after the mode change.

3.1.4 RoperScientific / Princeton

Intoduction

This plugin control a RoperScientific/Princeton camera under windows and linux, using the PVCAM (Photometrics Virtual Camera Access Method) libraries.

It is in production at SOLEIL under windows and it has been tested at Desy under linux . Model used at SOLEIL: PI-MTE:2048B

Prerequisite

The RoperScientific is connected to a specific computer with a PCI board the Lima/RoperScientific client has to Run on this PC

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

- The camera will be initialized within the RoperScientificCamera object. One should pass to the RoperScientificCamera constructor, the Camera number as an int.

for exemple : 0

Std capabilites

This plugin has been implemented in respect of the mandatory capabilites but with some limitations according to some programmer's choices. We only provide here extra information for a better understanding of the capabilities for the RoperScientific camera.

- HwDetInfo
- Max image size is : 2048 * 2048
- 16 bit unsigned type is supported
- HwSync

  trigger type supported are:

  - IntTrig
  - ExtTrigSingle
  - ExtTrigMult
  - ExtGate

Optional capabilites

- **HwBin**
  - all values are accepted
- HwRoi

Configuration

No Specific hardware configuration are needed

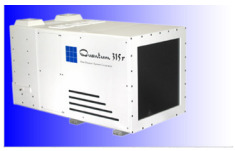
How to use

here is the list of accessible fonctions to configure and use the RoperScientific detector:

```
void setGain(long);
long getGain();

void setFullFrame(rgn_type* roi);
void setBinRoiParameters(rgn_type* roi);

void setSpeedTableIndex(unsigned);
unsigned getSpeedTableIndex(void);
const std::string& getADCRate(void);

double  getTemperature();
double  getTemperatureSetPoint();
void    setTemperatureSetPoint(double temperature);
```

3.2 Linux only

3.2.1 Adsc camera plugin



Introduction

ADSC : Area Detector System Corporation

The Lima module has been tested only with the 315r model:

Prerequisite

2 programs have to be running on the ADSC server: ccd_image_gather and det_api_workstation

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

Here are the available functions

- SetHeaderParameters
- UseStoredDarkImage
- SetImageKind
- SetLastImage

Std capabilites

This plugin has been implemented in respect of the mandatory capabilites but with some limitations according to some programmer's choices. We only provide here extra information for a better understanding of the capabilities for the Adsc camera.

- HwDetInfo
  - Max image size is : 3072 * 3072
  - 16 bit unsigned type is supported
- HwSync

  trigger type supported are:

  - IntTrig

Optional capabilites

- **HwBin**
  - 1 * 1
  - 2 * 2

Configuration

No Specific hardware configuration are needed

How to use

here is the list of accessible fonctions to configure and use the ADSC detector:

```
void    setHeaderParameters(const std::string& header);
void    setStoredImageDark(bool value);
bool    getStoredImageDark(void);
void    setImageKind(int image_kind);
int         getImageKind(void);
void    setLastImage(int last_image);
int         getLastImage(void);

void                setFileName(const std::string& name);
const std::string&  getFileName(void);
void                setImagePath(const std::string& path);
const std::string&  getImagePath(void);
```

3.2.2 Andor SDK2 camera plugin

Introduction

Andor Technology manufactuer offers a large catalogue of scientific cameras. Covered scientific applications are low ligth imaging, spectroscopy, microscopy, time-resolved and high energy detection. Andor is providing a unique

Software Development Tool (SDK) for both Windows and Linux, supporting different interface buses such as USB, CameraLink and also some specific acquisition PCI board.

The Lima module as been tested only with this cameras models:

- IKon-M (USB interface, Linux OS)

Prerequisite

Previously to this you have to install the Andor SDK the default path (/usr/local). For our test we us the SDK for Linux version **V2.91.30001.0** and ran the install script "install_andor" for which option 5 (All USB Cameras) was selected, the default installation is made under /usr/local/ with:

- /usr/local/include, header files
- /usr/local/lib, library files
- /usr/local/etc/andor, configuration files

The Linux SDK 2.91 has shared libraries which has been compiled on recent linux kernel, check first you have the right kernel and libc available by compiling one of the example program available under examples/console. Andor python module needs at least the lima core module.

For the USB camera the SDK is using the libusb under linux, check first your system is equiped with the libusb package otherwise you will not compile the Andor Lima plugin.

Installation & Module configuration

- follow first the steps for the linux installation *Installation*

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=0
COMPILE_ANDOR=1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_BASLER COMPILE_ANDOR COMPILE_CBF_SAVING
```

- start the compilation *Compilation*
- finally for the Tango server installation *Tango Device Server*

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

The camera will be initialized within the AndorCamera object. The AndorCamera contructor sets the camera with default parameters for Preampifier-Gain, VerticalShiftSpeed and the ADC/HorizontalSpeed.

These parameters are optimized for the faster mode, which means the maximum gain, the "fasten recommended" VSSpeed (i.e as returned by GetFastestRecommendedVSSpeed() SDK function call) and the ADC with the faster Horizontal speed.

All the parameters can be set and get using the corresponding methods, the default values (max speeds and gain) can be applied with -1 as passed value:

set/getPGain()

set/getVsSpeed()

set/getADCSpeed()

Std capabilites

This plugin has been implement in respect of the mandatory capabilites but with some limitations which are due to the camera and SDK features. We only provide here extra information for a better understanding of the capabilities for Andor cameras.

- HwDetInfo

  getCurrImageType/getDefImageType(): the methods call the SDK GetBitDepth() function to resolve the image data type. The bit-depth correspond to the AD channel dynamic range which depends on the selected ADC channel. By experience and with IKon detectors we only have Bpp16 of dynamic range, but the methods can return Bpp8 and Bpp32 as well.

  setCurrImageType(): this method do not change the image type which is fixed to 16bpp.

- HwSync

  get/setTrigMode(): the only supported mode are IntTrig, ExtTrigSingle, ExtGate and IntTrigMult

Optional capabilites

In addition to the standard capabilities, we make the choice to implement some optional capabilities which are supported by the SDK and the I-Kon cameras. A Shutter control, a hardware ROI and a hardware Binning are available.

- HwShutter

  setMode(): only ShutterAuto and ShutterManual modes are supported

- HwRoi

  There is no restriction for the ROI setting

- HwBin

  There is no restriction for the Binning but the maximum binning is given by the SDK function GetMaximumBinning() which depends on the camera model

Configuration

- Plug your USB camera on any USB port of the computer, that's all !!!

How to use

This is a python code example for a simple test:

```
from Lima import Andor
from lima impor Core

cam = Andor.Camera("/usr/local/etc/andor", 0)
hwint = Andor.Interface(cam)
control = Core.control(hwint)

acq = control.acquisition()

# configure some hw parameters
hwint.setTemperatureSP(-30)
hwint.setCooler(True)
.... wait here for cooling

hwint.setPGain(2)


# setting new file parameters and autosaving mode
saving=c.saving()

pars=saving.getParameters()
pars.directory='/buffer/lcb18012/opisg/test_lima'
pars.prefix='test1_'
pars.suffix='.edf'
pars.fileFormat=Core.CtSaving.EDF
pars.savingMode=Core.CtSaving.AutoFrame
saving.setParameters(pars)

# set accumulation mode

acq_pars= acq.getPars()

#0-normal,1-concatenation,2-accumu
acq_pars.acqMode = 2
acq_pars.accMaxExpoTime = 0.05
acq_pars.acqExpoTime =1
acq_pars.acqNbFrames = 1

acq.setPars(acq_pars)
# here we should have 21 accumalated images per frame
print acq.getAccNbFrames()

# now ask for 2 sec. exposure and 10 frames
acq.setAcqExpoTime(2)
acq.setNbImages(10)

acq.prepareAcq()
acq.startAcq()
```

3.2.3 Andor SDK3 camera plugin

Intoduction

CAUTION : This documentation is currently very partial and needs some extra work to be completed !!!

Andor Technology manufactuer offers a large catalogue of scientific cameras. Covered scientific applications are low ligth imaging, spectroscopy, microscopy, time-resolved and high energy detection. Andor is providing a Software Development Tool (SDK) for both Windows and Linux, supporting different interface buses such as USB, CameraLink and also some specific acquisition PCI board. Unfortunately there was a significant API change between the v2 line of SDK and the brand new v3 of the SDK, and recent cameras are only supported by the v3 SDK, whilst this new SDK is not (yet ?) supporting previously built cameras.

The Lima module as been tested only with this cameras models:

- Neo (3-tap Camera Link, Linux OS)

Module configuration

Previously to this you have to install the Andor SDK the default path (/usr/local). For our test we used the SDK for Linux version **V3.3.30004.0** and ran the install script "install_andor" for which option 2 (64b linux) was selected, the default installation is made under /usr/local/ with:

- /usr/local/include, header files
- /usr/local/lib, library files

- /usr/local/andor/bitflow, files for the frame-grabber driver (including camera firmware/frame grabber configuration)

The Linux SDK 3.3 has shared libraries which has been compiled on recent linux kernel, check first you have the right kernel and libc available by compiling one of the example program available under examples/console. Andor3 python module needs at least the lima core module.

The best before using this Lima pluggin with a Andor Neo camera is to test the proper setting of the frame-grabber driver and system configuration by using the two test programs included in the SDK. Those are typically found in /usr/local/andor/examples/ and are listdevices and image.

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=0
COMPILE_ANDOR=0
COMPILE_ANDOR3=1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_BASLER COMPILE_ANDOR COMPILE_ANDOR3 COMPILE_CBF_SAVING
```

See *Installation*

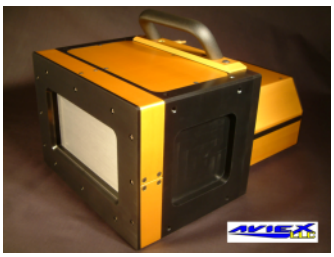
Installation

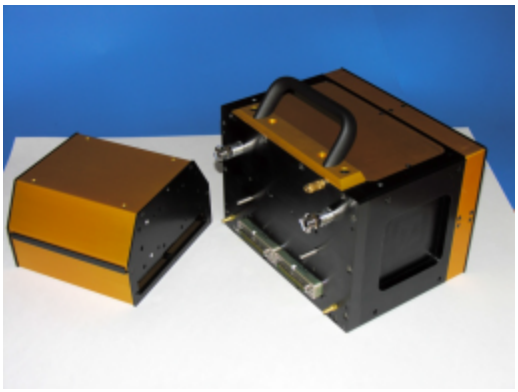
- After installing andor3 modules *Installation*
- And probably Tango server *Tango Device Server*

Configuration

- Still to be written.

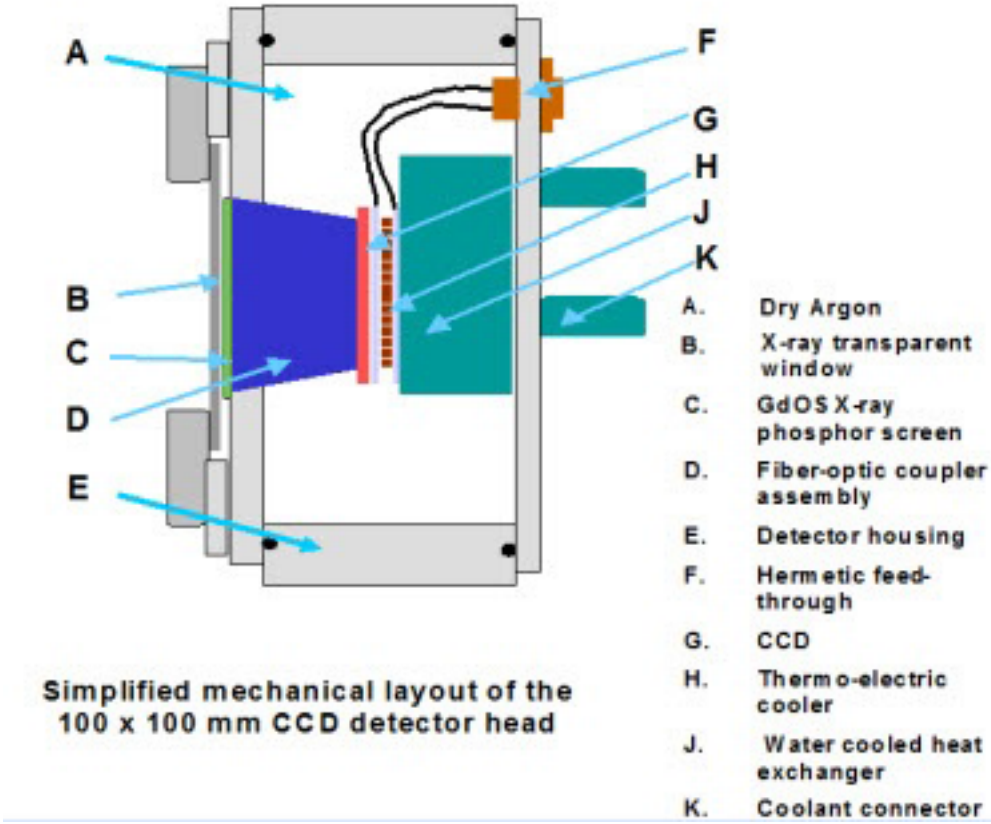
3.2.4 Aviex camera plugin



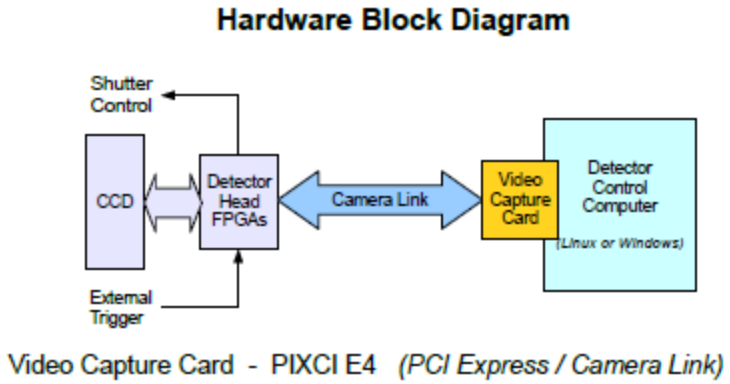


Intoduction

The PCCD-170170 is a large area detector (4096 x 4096) designed for use in WAXS or SAXS experiments in a vacuum environment.



Simplified mechanical layout of the 100 x 100 mm CCD detector head



The detector supports full frame, multiframe time-sliced, and streak camera modes of operation.

Used at the SWING beamline of Synchrotron SOLEIL to make timeresolved SAXS measurements together with another WAXS detector.

This Lima plugin controls an Aviex camera under linux.

It is based on the MX beamline control toolkit which can be found at http://mx.iit.edu

It has been tested at the Synchrotron SOLEIL facility, but is not already installed on a Beamline.

Module configuration

Previously to this you have to compile/install the Mx Library/Driver to the default path (/opt/mx/).

Starting the Mx driver doing this :

```
cd /opt/mx/sbin/
./mx start
```

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=0
COMPILE_AVIEX=1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_AVIEX COMPILE_PILATUS \
       COMPILE_BASLER COMPILE_CBF_SAVING
```

See *Installation*

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

There are 2 parameter to be filled by your Lima client:

- The detector friendly name , can be any string defined by user.
- The detector database file name : this file must contains configuration parameters as IP adress, port.

Std capabilites

This plugin has been implemented in respect of the mandatory capabilites but with some limitations according to some programmer's choices. We only provide here extra information for a better understanding of the capabilities for the Aviex camera.

- HwDetInfo
- Max image size is : 4096 * 4096
- 16 bit unsigned type is supported
- HwSync

  trigger type supported are:

  - IntTrig
  - ExtTrigSingle

Optional capabilites

- **HwBin**
  - 1 * 1
  - 2 * 2
  - 4 * 4
  - 8 * 8
  - ...

Binning above are typical values, but binning is not necessarily square.

- **HwRoi** Not yet implemented

Configuration

No Specific hardware configuration are needed

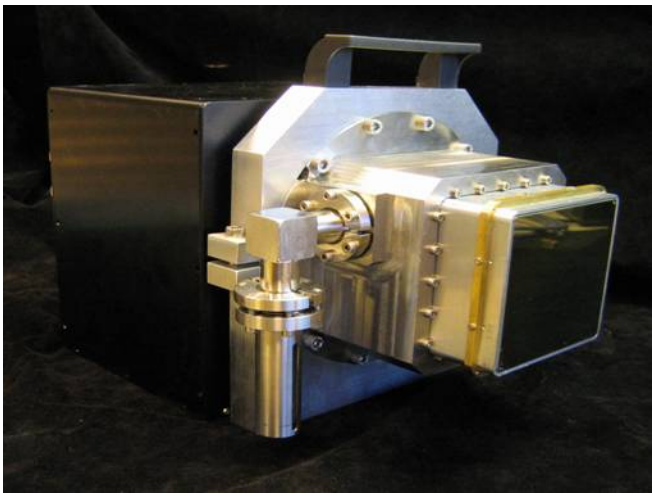
How to use

here is the list of accessible fonctions to configure and use the Aviex detector:

```
//-- Related to Aviex specific features
void getExpMultiplier(double& exp_mult);
void setExpMultiplier(double exp_mult);
void getLatencyTime(double& period_time);
void setLatencyTime(double  period_time);
```

```
void getGapMultiplier(double& gap_mult);
void setGapMultiplier(double  gap_mult);
void getMxLibraryVersion(std::string& version);
void getInternalAcqMode(std::string& acq_mode);
//! Available mode : ONESHOT, MULTIFRAME, GEOMETRICAL, MEASURE_DARK, MEASURE_FLOOD_FIELD
void setInternalAcqMode(const std::string& mode);
void getReadoutDelayTime(double& readout_delay);
void setReadoutDelayTime(double readout_delay);
void getReadoutSpeed(bool& readout_speed);
void setReadoutSpeed(bool readout_speed);
void getInitialDelayTime(double& initial_delay);
void setInitialDelayTime(double initial_delay);
//! MASK_CORRECTION_BIT_POSITION                = 0
//! BIAS_CORRECTION_BIT_POSITION                = 1
//! DARK_CORRECTION_BIT_POSITION                = 2
//! FLOOD_CORRECTION_BIT_POSITION               = 3
//! GEOM_CORRECTION_BIT_POSITION                = 12
void setCorrectionFlags(unsigned long);
```

3.2.5 Frelon camera plugin



Introduction

TODO

Prerequisite

Installation & Module configuration

- follow first the steps for the linux installation *Installation*

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
```

```
COMPILE_FRELON=1
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=0
COMPILE_PROSILICA=0
COMPILE_ROPERSCIENTIFIC=0
COMPILE_MYTHEN=0
COMPILE_ADSC=0
COMPILE_UEYE=0
COMPILE_XH=0
COMPILE_XSPRESS3=0
COMPILE_XPAD=0
COMPILE_PERKINELMER=0
COMPILE_ANDOR=0
COMPILE_PHOTONICSCIENCE=0
COMPILE_PCO=0
COMPILE_MARCCD=0
COMPILE_POINTGREY=0
COMPILE_IMXPAD=0
COMPILE_DEXELA=0
COMPILE_RAYONIXHS=0
COMPILE_CBF_SAVING=0
COMPILE_NXS_SAVING=0
COMPILE_FITS_SAVING=0
COMPILE_EDFGZ_SAVING=0
COMPILE_TIFF_SAVING=0
COMPILE_CONFIG=1
LINK_STRICT_VERSION=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
      COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
      COMPILE_BASLER COMPILE_PROSILICA COMPILE_ROPERSCIENTIFIC COMPILE_ADSC \
      COMPILE_MYTHEN COMPILE_UEYE COMPILE_XH COMPILE_XSPRESS3 COMPILE_XPAD COMPILE_PERKINELMER \
      COMPILE_ANDOR COMPILE_PHOTONICSCIENCE COMPILE_PCO COMPILE_MARCCD COMPILE_DEXELA\
      COMPILE_POINTGREY COMPILE_IMXPAD COMPILE_RAYONIXHS COMPILE_CBF_SAVING COMPILE_NXS_SAVING \
      COMPILE_FITS_SAVING COMPILE_EDFGZ_SAVING COMPILE_TIFF_SAVING COMPILE_CONFIG\
      LINK_STRICT_VERSION
```

- start the linux compilation *Compilation*
- finally with the Tango server installation *Tango Device Server*

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

The Frelon plugin is providing a helper class **FrelonAcq** which manages the initialisation sequence with the camera and interface object. One should only provide an Espia board channel number as the initialisation parameter (default is 0).

frelon = Frelon.FrelonAcq(int(espia_dev_nb)) return frelon.getGlobalControl()

Std capabilites

This plugin has been implemented in respect of the mandatory capabilites but with limitations according due to the detector specific features and with some programmer's choices. We do not explain here the standard Lima capabilites but you can find in this section the useful information on the Dexela specfic features.

- HwDetInfo

  TODO

- HwSync

  TODO

Optional capabilites

In addition to the standard capabilities, we make the choice to implement some optional capabilities in order to have an improved simulation.

- HwShutter

  TODO

- HwRoi

  TODO

- HwBin

  TODO

Configuration

The main configuration will consist in providing the correct "DexelaConfig.cfg" file to the detector API. The file has to be provided by the manufacturer with a second file like "sensor2923.fmt". The .fmt file contains some calibration data.

How to use

The LimaCCDs tango server provides a complete interface to the dexela plugin so feel free to test.

For a quick test one can use python, is this a short code example:

```
from Lima import Frelon
from lima impor Core
import time

FrelonAcq  = Frelon.FrelonAcq(int(espia_dev_nb))
control = FrelonAcq.getGlobalControl()

acq = control.acquisition()

# setting new file parameters and autosaving mode
saving=control.saving()

pars=saving.getParameters()
pars.directory='/tmp/'
pars.prefix='testfrelon_'
```

```
pars.suffix='.edf'
pars.fileFormat=Core.CtSaving.EDF
pars.savingMode=Core.CtSaving.AutoFrame
saving.setParameters(pars)


# now ask for 2 sec. exposure and 10 frames
acq.setAcqExpoTime(2)
acq.setNbImages(10)

acq.prepareAcq()
acq.startAcq()

# wait for last image (#9) ready
lastimg = control.getStatus().ImageCounters.LastImageReady
while lastimg !=9:
  time.sleep(1)
  lastimg = control.getStatus().ImageCounters.LastImageReady

# read a image
im0 = control.ReadImage(0)
```

3.2.6 Maxipix camera plugin

Intoduction

MAXIPIX is a high spatial resolution (small pixels), high frame rate, photon-counting pixel detector developed by ESRF. MAXIPIX is based on MEDIPIX2/TIMEPIX readout ASICs developed by CERN and the MEDIPIX2 collaboration. The active detector element consists of a hybrid pixel circuit glued on a chipboard and connected to it with microwire connections. The hybrid pixel circuit consists itself of a pixelated semiconductor sensor connected to one or several readout ASICs by individual micro solder bumps on each pixel. Various module formats are available and may implement either MEDIPIX2 or TIMEPIX ASICs. Both ASICs have identical pixel geometries but different characteristics as regards principally the lowest energy threshold, the discriminator range, and the available detection modes.

We provide today Maxipix 5x1, 4x1 and 1x1 formats based on both TIMEPIX and MEDIPIX2 ASICs.

Beamlines are equiped with the detector and its own control computer based on the famous Ecrin/Trenton 4U rackable PC.

Module configuration

Maxipix python module need at least the lima core module.

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=1
COMPILE_FRELON=0
COMPILE_MAXIPIX=1
COMPILE_PILATUS=0
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_CBF_SAVING
```

See *Installation*

3.2.7 DECTRIS EIGER 1M Detector System

Introduction

The EIGER 1M is a high performance X-Ray detector system.
It is made of two subsystems: a detector and a control server.
The control server is driven using an HTTP RESTful interface.
A C++ API for LImA has been developed at Synchrotron SOLEIL.

Prerequisite

Need to install some dependencies:

- libcurl

- liblz4
- libzmq
- libjsoncpp

On debian like system to install all dependencies, type this command:

```
seb@pcbliss02:~$ sudo apt-get install libcurl4-gnutls-dev liblz4-dev libzmq3-dev libjsoncpp-dev
```

Installation and Module configuration

- follow first the steps for the linux installation *Installation*

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=0
COMPILE_EIGER=1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_BASLER COMPILE_ANDOR COMPILE_CBF_SAVING
```

- start the compilation *Compilation*
- finally for the Tango server installation *Tango Device Server*

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialization

Initialization is performed automatically within the Eigercamera object. By default the stream will be use to retrieved images unless hardware saving is activated (CtSaving::setManagedMode(CtSaving::Hardware))

Std capabilities

- HwDetInfo

Capability	1M Value	4M Value	9M Value	16M Value
Maximum image size	1030 * 1065	2070 * 2167	3110 * 3269	4150 * 4371
Pixel depth	12 bits	12 bits	12 bits	12 bits
Pixel size	75µm * 75µm	75µm * 75µm	75µm * 75µm	75µm * 75µm
Maximum frame rate	3000Hz	750Hz	238Hz	133Hz

- **HwSync** Supported trigger types are:

- IntTrig
- IntTrigMult
- ExtTrigSingle
- ExtTrigMult
- ExtGate
- There is no hardware support for binning or roi.
- There is no shutter control.

Optional capabilities

- Cooling
- The detector uses liquid cooling.
- The API allows accessing the temperature and humidity as read-only values.

At the moment, the specific device supports the control of the following features of the Eiger Dectris API.
(Extended description can be found in the Eiger API user manual from Dectris).

- **Photon energy**: This should be set to the incoming beam energy. Actually it's an helper which set the threshold
- **Threshold energy**: This parameter will set the camera detection threshold. This should be set between 50 to 60 % of the incoming beam energy.
- Auto Summation (if activate image depth is 32 and, if not image depth is 16)
- HwSaving: This detector can directly generate hd5f, if this feature is used. Internally Lima control the file writer Eiger module. This capability can be activated though the control part with CtSaving object with setManagedMode method.
- Countrate correction
- Efficiency correction
- Flatfield correction
- LZ4 Compression
- Virtual pixel correction
- Pixelmask

Configuration

- Device configuration

The default values of the following properties must be updated in the specific device to meet your system configuration.

Property name	Description	Default value
DetectorIP	Defines the IP address of the Eiger control server (ex: 192.168.10.1)	127.0.0.1

3.2.8 Mythen

Intoduction

Server for the control of a Mythen detector.

Module configuration

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_ROPERSCIENTIFIC =0
COMPILE_MYTHEN =1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_ROPERSCIENTIFIC COMPILE_MYTHEN COMPILE_CBF_SAVING
```

Installation

Configuration

3.2.9 Pilatus

Intoduction

The PILATUS detector (pixel apparatus for the SLS) is a novel type of a x-ray detector, which has been developed at the Paul Scherrer Institut (PSI) for the Swiss Light Source (SLS). PILATUS detectors are two-dimensional hybrid pixel array detectors, which operate in single-photon counting mode. A hybrid pixel that features single photon counting, comprises a preamplifier, a comparator and a counter. The preamplifier enforces the charge generated in the sensor by the incoming x-ray; the comparator produces a digital signal if the incoming charge exceeds a predefined threshold and thus, together with the counter, one obtains a complete digital storage and read-out of the number of detected x-rays per pixel without any read-out noise or dark current!

PILATUS detectors feature several advantages compared to current state-of-the-art CCD and imaging plate detectors. The main features include: no readout noise, superior signal-to-noise ratio, read-out time of 5 ms, a dynamic range of 20bit, high detective quantum efficiency and the possibility to suppress fluorescence by a energy threshold that is set individually for each pixel. A more complete comparison is given in Table 1. The short readout and fast framing time allow to take diffraction data in continuous mode without opening and closing the shutter for each frame (see Fig. 1). For a comparison on the response to x-rays of integrating and single photon counting detectors see Fig. 2.

Because of the specified properties, PILATUS detectors are superiour to state-of-the-art CCD and imaging plate detectors for various x-ray detection experiments. Major improvements can be expected for time-resolved experiments, for the study of weak diffraction phenomena (e.g. diffuse scattering), for accurate measurements of Bragg intensities, for resonant scattering experiments,...

Module configuration

Pilatus python module need at least the lima core module.

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_CBF_SAVING
```

See *Installation*

Installation

- After installing python modules on pilatus pc *Installation*
- And probably Tango server *Tango Device Server*
- On Pilatus PC create **as root** a ramdisk of 8GB which will be used by Lima dserver as temporary buffer:
  - edit file /etc/fstab and add the following line:

```
none                    /lima_data tmpfs    size=8g,mode=0777      0 0
```

  - make the directory:

```
mkdir /lima_data
```

  - and finally mount the ramdisk:

```
mount -a
```

- For Pilatus3 edit file ~det/p2_det/config/cam_data/camera.def and add thoses two lines:
  - camera_wide = WIDTH_OF_THE_DETECTOR
  - camera_high = HEIGHT_OF_THE_DETECTOR

Start the system

- Log on the detector pc as *det* user start tvx/camserver:

```
cd p2_det
./runtvx
```

- when tvx has finished initializing camserver just type *quit* in tvx window
- Log on the detector pc as *an other user* or *det*

```
cd WHERE_YOU_HAVE_INSTALL_PILATUS_TANGO_SERVER
TANGO_HOST=Host:Port python LimaCCD.py instance_name
```

If the cameserver window notice a connection, seams to work ;)

3.2.10 imXPAD camera plugin

Introduction

The imXpad detectors benefit of hybrid pixel technology, which leads to major advantages compared to the other technologies. These advantages are mainly provided by direct photon conversion and real time electronic analysis of X-ray photons. This allows for direct photon counting and energy selection.

XPAD detectors key features compared to CCDs and CMOS pixels detectors are: - Noise suppression - Energy selection - Almost infinite dynamic range - High Quantum Efficiency (DQE(0) ~100%, dose reduction) - Ultra fast electronic shutter (10 ns) - Frame rate > 500 Hz

Prerequisite

In order to operate the imXpad detector, the USB-server or the PCI-server must be running in the computer attached to the detector.

Installation & Module configuration

- follow first the steps for the linux installation *Installation*

The minimum configuration file is *config.inc*:

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_IMXPAD=1
COMPILE_CONFIG=1
COMPILE_GLDISPLAY=0
LINK_STRICT_VERSION=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
     COMPILE_IMXPAD \
```

```
COMPILE_GLDISPLAY \
LINK_STRICT_VERSION
```

- start the compilation *Compilation*
- finally for the Tango server installation *Tango Device Server*

Initialisation and Capabilities

Camera initialisation

imXpad camera must be initialisated using 3 parameters:

1. The IP adress where the USB or PCI server is running
2. The port number use by the server to communicate.
3. The detector model.

Std capabilites

- HwDetInfo

  getCurrImageType/getDefImageType():

- HwSync:

  get/setTrigMode(): the only supported mode are IntTrig, ExtTrigSingle, ExtGate.

Refer to: http://80.15.49.71/manuals.html for a whole description of detector capabilities.

Optional capabilites

This plugin does not use optional hardware capabilities.

How to use

This is a python code example for a simple test:

```
from Lima import imXpad
from Lima import Core

cam = imXpad.Camera('10.211.55.2', 3456, 'XPAD_S70')
HWI = imXpad.Interface(cam)
CT = Core.CtControl(HWI)
CTa = CT.acquisition()
CTs = CT.saving()
CTs.setDirectory("/home/imxpad/data")
CTs.setPrefix("20140515_01_")
CTs.setFormat(CTs.EDF)
CTs.setSuffix(".edf")
CTs.setSavingMode(CTs.AutoFrame)
CTa.setAcqExpoTime(1)
CTa.setAcqNbFrames(5)
cam.init()
```

```
CT.prepareAcq()
CT.startAcq()
cam.stopAcq()
cam.exit()
```

3.2.11 PointGrey

Intoduction

Poiint Grey is a world-leading designer and manufacturer of innovative, high-performance digital cameras for industrial, life science, and traffic applications. We offer a unique and comprehensive portfolio of USB 3.0, GigE, FireWire, USB 2.0 and Camera Link products known for their outstanding quality, ease of use, and unbeatable price-performance.

The Lima module as been tested only with this GigE cameras models:

- Blackfly 1024x768 (model BFLY-PGE-05S2M)

Prerequisite

Previously to this you have to install the PointGrey *FlyCapture* SDK. We only tested it on debian6 and using the SDK version 2.3.19 (the latest one compatible with debian6 libc).

PointGrey python module need at least the lima core module.

Installation & Module configuration

- follow first the steps for the linux installation *Installation*

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_POINTGREY=1
COMPILE_BASLER=0
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_POINTGREY COMPILE_BASLER COMPILE_CBF_SAVING
```

- start the compilation *Compilation*
- finally for the Tango server installation *Tango Device Server*

Initialisation and Capabilities

Implementing a new plugin for new detector is driven by the LIMA framework but the developer has some freedoms to choose which standard and specific features will be make available. This section is supposed to give you good knowledge regarding camera features within the LIMA framework.

Camera initialisation

The camera has to be initialized using the PointGreyCamera class. The default constructor needs at least the serial number of your camera in order to get the network connection setting up. In Addition one can provide both packate_size and packet_delay parameters. By default no value is passed.

Std capabilities

This plugin has been implement in respect of the mandatory capabilites but with some limitations which are due to the camera and SDK features. We only provide here extra information for a better understanding of the capabilities for Andor cameras.

- HwDetInfo

  getPixelSize(): the method just returns -1, it has to be implemented in further version. get/setImageType(): the plugin only supports Bpp8 and Bpp16

- HwSync

  get/setTriggerMode(): Depending of the camera model, but some can not support any trigger mode. Otherwise the only implemented modes are IntTrig and ExtTrigSingle. IntTrigMult is normally a mandatory mode (for any camera) and will be implemented in next version.

Optional capabilities

None has been implemented for this camera plugin.

Specific control parameters

Some specific paramaters are available within the camera hardware interface. Those parameters should be used carefully and one should refer to the camera SDK (or user's guide) documentation for a better understanding.

- get/setPacketSize()
- get/setPacketDelay()
- get/setGain()
- get/setAutoGain()
- getGainRange()

The following parameters can break the synchronisation with the LIMA HwSync layer by changing the camera internal exposure time.

- get/setAutoExpTime()
- get/setFrameRate()
- get/setAutoFrameRate()

Network Configuration

- Depending on your network infrastructure you will need to configure a fix IP address for the camera or use a DHCP setup instead.

  The linux SDK provides a configuation tool called *GiGEConfigCmd*. The Windows SDK version provides a graphical tool, *GigEConfigurator.exe*

- Then in the PointGrey Tango device set the property *camera_serial* using the camera serial number (sticked on the camera).
- If you are running the server with linux kernel >= 2.6.13, you should add this line into *etc/security/limits.conf*. With this line, the acquisition thread will be in real time mode.

```
USER_RUNNING_DEVICE_SERVER    -       rtprio  99
```

How to use

This is a python code example for a simple test:

```
from Lima import PointGrey
from lima impor Core

cam = PointGrey.Camera(13125072)
hwint = PointGrey.Interface(cam)
control = Core.control(hwint)

acq = control.acquisition()

# configure some hw parameters
```

```
hwint.setAutoGain(True)
hwint.se

# setting new file parameters and autosaving mode
saving=c.saving()

pars=saving.getParameters()
pars.directory='/buffer/lcb18012/opisg/test_lima'
pars.prefix='test1_'
pars.suffix='.edf'
pars.fileFormat=Core.CtSaving.EDF
pars.savingMode=Core.CtSaving.AutoFrame
saving.setParameters(pars)

# now ask for 10ms sec. exposure and 100 frames
acq.setAcqExpoTime(0.01)
acq.setNbImages(100)

acq.prepareAcq()
acq.startAcq()
```

3.2.12 Prosilica

Introduction

AVT offers a large choice of FireWire and GigE cameras for machine vision, computer vision and other industrial or medical applications. Cameras by AVT and Prosilica include sensitive machine vision sensors (CCD and CMOS, VGA to 16 Megapixels) and fit a wide range of applications.

The Lima module as been tested with color and B/W GigE camera.

Module configuration

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=0
COMPILE_PROSILICA=1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_BASLER COMPILE_PROSILICA COMPILE_CBF_SAVING
```

See *Installation*

Installation

- After installing prosilica modules *Installation*
- And probably Tango server *Tango Device Server*

Configuration

- First you have to setup ip addresse of the Prosilica Camera by using *CLIpConfig* (camera/prosilica/sdk/CLIpConfig)
- list of all cameras available : *CLIpConfig -l* (If you do not see any camera, that bad news!)
- finally set ip add : *CLIpConfig -u UNIQUE_NUMBER -s -i 169.254.X.X -n 255.255.255.0 -m FIXED* (It's an example!)
- Then in the Prosilica Tango device set the property *cam_ip_address* to the address previously set.

That's all....

3.2.13 MarCCD

Intoduction

The SX165 features a round, 165 mm diameter active area, and a versatile, high resolution CCD chip. It is the ideal X-ray detector for research applications with both synchrotrons and rotating anode X-ray sources.

Prerequisite

The MarCCD software server should be started on the MarCCD host computer, by running the command:

```
marccd -r
```

Then you can launch your lima/marccd client on another host, as the MarCCD server can be reached by network

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

There are 4 parameter to be filled by your Lima client:

- The IPAddress of the host where the marccd server is running
- The port of the marccd server process
- The detector target path: the path where will be saved the marccd image files
- Reader timeout: in ms, the timeout after which the plugin will be in fault if no marccd image file is present

Std capabilites

This plugin has been implemented in respect of the mandatory capabilites but with some limitations according to some programmer's choices. We only provide here extra information for a better understanding of the capabilities for the MarCCD camera.

- HwDetInfo
- Max image size is : 4096 * 4096
- 16 bit unsigned type is supported
- HwSync

  trigger type supported are:

  - IntTrig

Optional capabilites

- **HwBin**
  - 2 * 2
  - 4 * 4
  - 8 * 8

- HwRoi

Configuration

No Specific hardware configuration are needed

How to use

here is the list of accessible fonctions to configure and use the MarCCD detector:

```
void getDetectorImageSize(Size& size);
void setImagePath(const std::string& path);
const std::string& getImagePath(void);
void setImageFileName(const std::string& imgName);
const std::string& getImageFileName();
void setImageIndex(int newImgIdx);
int getImageIndex();
int getFirstImage();

bool isStopSequenceFinished();
void saveBGFrame(bool);

void setBeamX(float);
float getBeamX();
void setBeamY(float);
float getBeamY();
void setDistance(float);
float getDistance();

void setWavelength(float);
float getWavelength();
```

3.2.14 Rayonix HS

rayonix High-performance X-ray technology

Intoduction

The MX-HS series from Rayonix incorporates the new, exclusive HS frame-transfer technology for high speed X-ray data collection without compromising resolution or data quality. The result is a new type of high speed and ultra-low noise area detector that delivers the highest performance available for X-ray diffraction applications.

The Rayonix MX-HS detectors are ideal for taking advantage of high brilliance synchrotron sources, or for any other high frame rate application. Examples include: high throughput protein crystallography, Laue diffraction, time-resolved or static small-angle X-ray scattering (SAXS), wide-angle X-ray scattering (WAXS), powder diffraction,

X-ray computed tomography (CT), X-ray imaging, and coherent diffraction imaging (CDI). With no count rate limitation, these detectors are also ideal for XFEL applications.

The Lima module as been tested only with the following models :

- MX170-HS (2x2 mdules)

Prerequisite

The Rayonix HS detector is been delivered today wih its own control computer, a powerful computer embedded at leat 8GB of RAM, dual 4-Core CPU (8 cores) and a GPU card for the online image correction (background, flatfield ...). The computer is running redhat entprise Linux 6 (64bits).

The rayonix DSK is preinstalled on the detector node under the directory **/opt/rayonix**.

There is no special prerequisite then you can test that everything is OK with installation by running the rayonix GUI, **caxpure**.

Installation & Module configuration

- follow first the steps for the linux installation *Installation*

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_POINTGREY=0
COMPILE_BASLER=0
COMPILE_RAYONIXHS=1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_POINTGREY COMPILE_RAYONIXHS COMPILE_BASLER COMPILE_CBF_SAVING
```

- start the compilation *Compilation*
- finally for the Tango server installation *Tango Device Server*

Initialisation and Capabilities

Implementing a new plugin for new detector is driven by the LIMA framework but the developer has some freedoms to choose which standard and specific features will be make available. This section is supposed to give you the correct information regarding how the camera is exported within the LIMA framework.

Camera initialisation

The camera has to be initialized using the RayonixHsCamera class. The default constructor does not need any input parameter.

Std capabilities

This plugin has been implement in respect of the mandatory capabilites but with some limitations which are due to the camera and SDK features. We only provide here extra information for a better understanding of the capabilities.

- HwDetInfo

  The detector is set to full image size at startup which means a binning of 1x1. The recommended binning for most of the experiment is 2x2.

- HwSync
- The minimum latency time is 1 ms.
- The supported trigger modes are depending of the chosen frame mode. There are:
  - IntTrig
  - IntTrigMult
  - ExtTrigSingle
  - ExtTrigMult (only for SINGLE frame mode)
  - ExtGate (only for SINGLE frame mode)
  - ExtTrigReadout (only for FAST_TRANSFER frame mode).

Optional capabilities

- HwBin

The supported hardware binning are 2x2, 3x3, 4x4, 5x5, 6x6, 7x7, 8x8, 9x9 and 10x10. By increasing the binning factor you can increase the readout speed from 2.6 fps to 140 fps which corresponds respectively to a pixel size of 44um and 440 um.

- HwShutter

The Rayonix HS detectors provides 2 output channels one can choose a different source for each (see specific control parameters for more details about the output source control). For the SHUTTER source both opening and closing delay can be set. The Rayonix HS shutter capability only supports two modes:

- **ShutterAutoFrame**
- **ShutterManual**

Specific control parameters

Some specific paramaters are available within the camera hardware interface. Those parameters should be used carefully and one should refer to the camera SDK (or user's guide) documentation for a better understanding.

- get/setFrameTriggerType(type): signal type for the frame trigger input (channel #1)
- get/setSequenceGateSignalType(type): signal type for the gate input (channel #2), The supported signal types:
- OPTO
- OPTO_INVERTED
- CMOS
- CMOS_PULLDOWN

- CMOS_PULLUP
- CMOS_PULLDOWN_INVERTED
- CMOS_PULLUP_INVERTED
- SOFTWARE
- get/setOutputSignalType(channel, type): the signal type for the output channel (CHANNEL_1 or CHANNEL_2)
- get/setOutputSignalID(channel, id): the source id for the output channel, possible sources are:
  - ID_SHUTTER
  - ID_INTEGRATE
  - ID_FRAME
  - ID_LINE
  - ID_SHUTTER_OPENING
  - ID_SHUTTER_CLOSING
  - ID_SHUTTER_ACTIVE
  - ID_TRIGGER_RISE_WAIT
  - ID_TRIGGER_RISE_ACK
  - ID_TRIGGER_FALL_WAIT
  - ID_TRIGGER_FALL_ACK
  - ID_TRIGGER_2_RISE_WAIT
  - ID_TRIGGER_2_RISE_ACK
  - ID_INPUT_FRAME
  - ID_INPUT_GATE
- get/setElectronicShutterEnabled(): active or unactive the electronic shutter
- get/setCoolerTemperatureSetpoint(): the cooler temperature set-point
- get/setSensorTemperatureSetpoint(): the sensor temperature set-point
- get/setSensorTemperature(): the detector measured temperature
- get/setCooler(): stop or start the cooler controller
- get/setVacuumValve(): close or open the vacuum valve
- get/setFrameMode(): modes are SINGLE or FAST_TRANSFER.

  WARNING: in FAST_TRANSFER mode the latency time is disabled and it has a fixed value of 1 ms which corresponds to the readout time. In addition to this the supported trigger mode will depend on the frame mode. The list of supported trigger modes is available in this document below.

Configuration

- Cabling

  The detector head should be connected to the detector computer on the cameralink and USB links. You must connect the USB on the PCI board (not the motherboard ones) and the cameralink on the first channel, the top connector.

- Cooling

  For an optimized condition wit dark current the detector has to be cooled down, the sensor temperature set-point should be at -120 deg and the cooler temperature set-point at -90 deg Celsuis. And of course the cooler controller should be started.

How to use

This is a simple python test program:

```
from Lima import RayonixHs
from lima impor Core

cam = RayonixHs.Camera()
hwint = RayonixHs.Interface(cam)
control = Core.control(hwint)

acq = control.acquisition()

# configure some hw parameters
sens_temp = hwint.getSensorTemperature()
cool_temp = hwint.getCoolerTemperatureSetpoint()
if sens_temp > -50:
  print " Hoops, detector is not cooled down, temp = ", sens_temp

# setting new file parameters and autosaving mode
saving=c.saving()

pars=saving.getParameters()
pars.directory='/somewhere/'
pars.prefix='test1_'
pars.suffix='.edf'
pars.fileFormat=Core.CtSaving.EDF
pars.savingMode=Core.CtSaving.AutoFrame
saving.setParameters(pars)

# set a new binning to increase the frame rate
image = control.image()
image.setBin(Core.Bin(2,2))

# now ask for 10ms sec. exposure and 100 frames
acq.setAcqExpoTime(0.01)
acq.setNbImages(100)

acq.prepareAcq()
acq.startAcq()
```

3.2.15 Ueye

Intoduction

Industrial Cameras for digital imaging and visualization (USB,GigE).

home site: http://www.ids-imaging.com/

Module configuration

Previously to this you have to install the Ueye SDK

Ueye python module need at least the lima core module.

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=0
COMPILE_PROSILICA=0
COMPILE_ROPERSCIENTIFIC=0
COMPILE_MYTHEN=0
COMPILE_ADSC=0
COMPILE_UEYE=1
COMPILE_CBF_SAVING=0
COMPILE_NXS_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
     COMPILE_BASLER COMPILE_PROSILICA COMPILE_ROPERSCIENTIFIC COMPILE_ADSC \
     COMPILE_MYTHEN COMPILE_UEYE COMPILE_CBF_SAVING COMPILE_NXS_SAVING
```

See *Installation*

Installation

- After installing ueye modules *Installation*
- And probably Tango server *Tango Device Server*

Configuration

See the sdk README provide in the ueye module

3.2.16 Ultra

Introduction

The ULTRA Detector System enables capture of one dimensional spectra at extremely high rates. Where CCDs were used to capture a line of data at a time, the ULTRA Detector System offers many orders of magnitude faster time framing. ULTRA is a compact turnkey system. The data acquisition system is attached in a compact form factor unit with gigabit Ethernet out and multiple I/O options onboard.

Ultra Specification:-

Sustained Spectral Rate 20 KHz (spectra per second) Maximum Frame Period <500 ns Minimum Spectral Sensitivity 5 – 17KeV 300µm thickness. 500µm also available. Output Gigabit Ethernet Pixel configuration Si 512 linear strips @ 50µm pitch ADC Dynamic Range 16 Bit Synchronisation Inputs TTL or Fibre Optic Integration Time <1us - 650us frames TriggeringExternal (TTL or Fibre) or Internal (10KHz fixed)

Prerequisite

The default network setup is (excluding the site network connection):

1GBit Copper network for control communinication between the PC and the Ultra box.

Installation & Module configuration

- follow first the steps for the linux installation *Installation*

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ULTRA=1
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR COMPILE_ULTRA
```

- start the compilation *Compilation*
- finally for the Tango server installation *Tango Device Server*

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

The camera will be initialized within the UltraCamera object. A TCP and UDP socket connections on the 1GBit port are established

The Ultra requires the following parameters with the recommended settings. headname = 192.168.1.100 hostname = 192.168.1.103 tcpPort = 7 udpPort = 5005 npixels = 512

Std capabilites

This plugin has been implemented with respect of the mandatory capabilites but with some limitations which are due to the camera. We only provide here extra information for a better understanding of the capabilities for Ultra cameras.

- HwDetInfo

  getCurrImageType/getDefImageType(): is set to Bpp16

- HwSync

  get/setTrigMode(): the only supported modes are IntTrig, ExtTrigMult and IntTrigMult

Optional capabilities

3.2.17 Xpad

Intoduction

The XPAD detector is based on the photon counting technology providing a quasi noiseless imaging as well as a very high dynamic range and a fast frame rate (500 images/s). This is a detector from the collaboration of Soleil, CPPM and ESRF(D2AM). It is now supported by the ImXPAD company. This plugin support the folling models: S70,S140,S340,S540

The XPAD run under Linux, with the help of a PCI express board from PLDA.

Module configuration

There is no specific module configuration, as the host is provided by the manufacturer

3.2.18 Xspress3

Introduction

Many solid state detectors are not limited by their intrinsic rate capability, but by the readout system connected to them. The Quantum Detectors Xspress 3 was developed to maximise the throughput and resolution of such detectors

and remove the bottleneck at the readout stage. With output count rates of over 3 Mcps, this detector is easily 10X faster than the systems many users have on their beamlines. Xspress 3 can open up the beamline to much faster data collection, its dynamic range can reduce the number of scans required and save large amounts of time with attenuation selection.

The XSPRESS3 system contains a Xilinx Virtex-5 FPGA with two embedded PowerPC processors. PPC1 manages the DMA engines. PPC2 runs the Xilinx micro kernel and communicates to the Intel 64 bit Linux server PC by 1 GBit Ethernet,TCP sockets. Bulk data and event lists to be histogrammed are sent from the firmware to the Server PC by 10G Ethernet, UDP.

The Software Development Toolkit (SDK) is provided for Linux only.

Prerequisite

Unpack the SDK distribution into either the camera/xspress3/sdk directory or /usr/local/lib Then ensure the libraries are in the LD_LIBRARY_PATH

The SDK has shared libraries which has been compiled on recent linux kernel. g++ (GCC) 4.1.2 20080704 (Red Hat 4.1.2-50), check first you have the right kernel and libc available by compiling the test program.

The default network setup is (excluding the site network connection):

1GBit Copper network for control communinication between the PC and the XSPRESS3 box. With more than 1 XSPRESS3 box connected this network uses a ethernet switch A private network with 64 addresses allocated: eth1 Link encap:Ethernet HWaddr d4:ae:52:7d:5f:84

> inet addr:192.168.0.1 Bcast:192.168.0.63 Mask:255.255.255.192 inet6 addr: fe80::d6ae:52ff:fe7d:5f84/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:9000 Metric:1 RX packets:1567 errors:0 dropped:5766 overruns:0 frame:0 TX packets:158 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:173937 (169.8 KiB) TX bytes:37252 (36.3 KiB) Interrupt:48 Memory:da000000-da012800

A 10GBit Fibre network for data transfer, point to point with 4 addresses allocated. With more that 1 XSPRESS3 box there would be multiple 10G Ports on the PC with multiple 4 address range subnets

ifconfig eth2 eth2 Link encap:Ethernet HWaddr 00:07:43:05:7c:65

> inet addr:192.168.0.65 Bcast:192.168.0.67 Mask:255.255.255.252 inet6 addr: fe80::207:43ff:fe05:7c65/64 Scope:Link UP BROADCAST RUNNING MULTICAST MTU:9000 Metric:1 RX packets:0 errors:0 dropped:0 overruns:0 frame:0 TX packets:702 errors:0 dropped:0 overruns:0 carrier:0 collisions:0 txqueuelen:1000 RX bytes:0 (0.0 B) TX bytes:154963 (151.3 KiB) Interrupt:41 Memory:dd7fe000-dd7fefff

Note the carefully picked subnet masks etc and the MTU 9000 We then have a script that should be executed automatically at boot.

cat /etc/init.d/xspress3.sh

```
#!/bin/bash
#
# static-arp        This is to register a static ARP address in the arp table at boot
#
# Kept as simple as possible hopefully this will auto register the associated
# MAC with the private network address to allow the machine to communicate with the
# test boards for xspress3
# Derived from work by Duncan Russell, by William Helsby
PATH=/sbin:/bin:/usr/bin:/usr/sbin
arp -i eth2 -s 192.168.0.66 02:00:00:00:00:00
#route -v add -host 192.168.0.66 eth2
# Setting default and max buffer sizes for networking.
```

```
sysctl -w net.core.rmem_max=1073741824
sysctl -w net.core.rmem_default=1073741824
```

Installation & Module configuration

- follow first the steps for the linux installation *Installation*

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_XSPRESS3=1
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR COMPILE_XSPRESS3
```

- start the compilation *Compilation*
- finally for the Tango server installation *Tango Device Server*

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

The camera will be initialized within the Xspress3Camera object. A TCP socket connection on the 1GBit port is established and optionally a UDP connection on the 10Gbit port (depends on boolean constructor flag noUDP). The ROI's are reset, the first card in a multicard system or the single card, is set to be the master and the run flags are set to initiate Scaler and Histogram modes. The register and configuration settings (as optimised by QD on delivery) are uploaded to the Xspress3.

The Xspress3 requires the following parameters with the recommended settings. nbCards = 1 (number of Xspress3 boxes) maxFrames = 16384 baseIPaddress = "192.168.0.1" basePort = 30123 baseMACaddress = "02.00.00.00.00.00" nbChans = 4/6/8 (depends on the firmware) createScopeModule = true/false scopeModuleName = "a-name-of-your-choice" debug = 0 is off, 1 is on, 2 is verbose cardIndex = 0 (for a 1 xspress system) noUDP = true/false directoryName = "directory containing xspress3 configuration settings"

The Xspress3Camera contructor sets the camera with default parameters for Number of Pixels (4096), the imageType (Bpp32), Number of Frames (1) and the trigger mode (IntTrig)

Std capabilites

This plugin has been implemented with respect of the mandatory capabilites but with some limitations which are due to the camera and SDK features. We only provide here extra information for a better understanding of the capabilities for Xspress3 cameras.

- HwDetInfo

  getCurrImageType/getDefImageType(): is set to Bpp32 setCurrImageType(): will not change the image type. getMaxImageSize/getDetectorImageSize(): is defined as number of pixels + number of scalers x number of channels.

  i.e. (4096+8) x 4 for a 4 channel xspress3 system

getPixelSize(): is hardcoded to be 1x1 getDetectorModel(): reads and reports the xspress3 firmware version.

- HwSync

  get/setTrigMode(): the only supported modes are IntTrig, ExtGate and IntTrigMult

Optional capabilities

None

Data Format

The raw data is saved in .edf file format. Each frame is saved as it completes. To allow Lima to save both histogram and scaler data, the latter is appended to the histogram data.

histogram scaler

[0] [0 ... 4095, 4096 ... 5003] channel 0 [1] [0 ... 4095, 4096 ... 5003] channel 1 [2] [0 ... 4095, 4096 ... 5003] channel 2 [3] [0 ... 4095, 4096 ... 5003] channel 3

Camera::readScalers(): returns the raw scaler data from the Lima buffers from the specified frame and channel Camera::readHistogram(): returns the raw histogram data from the Lima buffers from the specified frame and channel setUseDtc/getUseDtc(): set to true will dead time correct the data returned from the Lima buffers (default is false) setUseHW/getUseHw(): set to true will return raw histogram data from the H/W data buffers, including the current frame.

How to use

See example in the test directory. Playback data should be extracted from the tarball.

3.3 Windows and Linux

3.3.1 Basler camera plugin

Intoduction

Basler's area scan cameras are designed for industrial users who demand superior image quality and an excellent price/performance ratio. You can choose from an area scan portfolio that includes monochrome or color models with various resolutions, frame rates, and sensor technologies.

The Lima module as been tested only with this GigE cameras models:

- Scout
- Pilot
- Ace

And for now It's only work with monochrome cameras

Module configuration

Previously to this you have to install the Basler SDK *Pylon* to the default path (/opt/pylon)

Basler python module need at least the lima core module.

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=0
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
COMPILE_PILATUS=0
COMPILE_BASLER=1
COMPILE_CBF_SAVING=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
       COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
       COMPILE_BASLER COMPILE_CBF_SAVING
```

See *Installation*

Installation

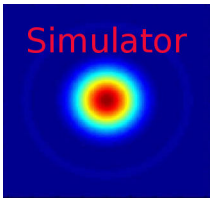
- After installing basler modules *Installation*
- And probably Tango server *Tango Device Server*

Configuration

- First you have to setup ip addresse of the Basler Camera by using *IpConfigurator* (/opt/pylon/bin/IpConfigurator)
- Then in the Basler Tango device set the property *cam_ip_address* to the address previously set.
- If you are running the server with linux kernel >= 2.6.13, you should add this line into *etc/security/limits.conf*. With this line, the acquisition thread will be in real time mode.

```
USER_RUNNING_DEVICE_SERVER    -       rtprio  99
```

3.3.2 Simulator



Introduction

This is the official Lima camera simulator. It has been made to help you starting with Lima and to test/play Lima without any hardware. The simulator is able to provide frames with diffraction patterns and a set of parameters can be tuned to change those patterns like for instance the number and position of peaks (i.e. FrameBuilder class).

Prerequisite

There is no special prerequisite, one can compile and test it on both linux and windows platforms.

Installation & Module configuration

- follow first the steps for the linux installation *Installation*
- follow first the steps for the windows installation *GetIt*

The minimum configuration file is *config.inc* :

```
COMPILE_CORE=1
COMPILE_SIMULATOR=1
COMPILE_SPS_IMAGE=1
COMPILE_ESPIA=0
COMPILE_FRELON=0
COMPILE_MAXIPIX=0
```

```
COMPILE_PILATUS=0
COMPILE_BASLER=0
COMPILE_PROSILICA=0
COMPILE_ROPERSCIENTIFIC=0
COMPILE_MYTHEN=0
COMPILE_ADSC=0
COMPILE_UEYE=0
COMPILE_XH=0
COMPILE_XSPRESS3=0
COMPILE_XPAD=0
COMPILE_PERKINELMER=0
COMPILE_ANDOR=0
COMPILE_PHOTONICSCIENCE=0
COMPILE_PCO=0
COMPILE_MARCCD=0
COMPILE_POINTGREY=0
COMPILE_IMXPAD=0
COMPILE_DEXELA=0
COMPILE_RAYONIXHS=0
COMPILE_CBF_SAVING=0
COMPILE_NXS_SAVING=0
COMPILE_FITS_SAVING=0
COMPILE_EDFGZ_SAVING=0
COMPILE_TIFF_SAVING=0
COMPILE_CONFIG=1
LINK_STRICT_VERSION=0
export COMPILE_CORE COMPILE_SPS_IMAGE COMPILE_SIMULATOR \
      COMPILE_ESPIA COMPILE_FRELON COMPILE_MAXIPIX COMPILE_PILATUS \
      COMPILE_BASLER COMPILE_PROSILICA COMPILE_ROPERSCIENTIFIC COMPILE_ADSC \
      COMPILE_MYTHEN COMPILE_UEYE COMPILE_XH COMPILE_XSPRESS3 COMPILE_XPAD COMPILE_PERKINELMER \
      COMPILE_ANDOR COMPILE_PHOTONICSCIENCE COMPILE_PCO COMPILE_MARCCD COMPILE_DEXELA\
      COMPILE_POINTGREY COMPILE_IMXPAD COMPILE_RAYONIXHS COMPILE_CBF_SAVING COMPILE_NXS_SAVING \
      COMPILE_FITS_SAVING COMPILE_EDFGZ_SAVING COMPILE_TIFF_SAVING COMPILE_CONFIG\
      LINK_STRICT_VERSION
```

- start the compilation *Compilation*
- finally for the Tango server installation *Tango Device Server*

Initialisation and Capabilities

In order to help people to understand how the camera plugin has been implemented in LImA this section provide some important information about the developer's choices.

Camera initialisation

The camera will be initialized within the SimulatorCamera object. There is no parameters to pass to the Simulato-Camera() constructor.

This simulator plugin is based on the use of a frame builder class called **FrameBuidler** (SimulatorFrameBuilder.h) on which you can apply few different settings. The SimulatorCamera class provides a special method to get the **FrameBuilder** instance (SimulatorCamera::getFrameBuilder()). Then one can play with few parameters like:

- setFrameDim(): set a new frame dimension (max. is 1024x1024)
- setPeaks(): set a list of GaussPeak positions (GaussPeak struct -> x, y, fwhm, max)

- setPeakAngles(): set a list of GaussPeak angles
- setFillType(): set the image fill type Gauss or Diffraction (default is Gauss)
- setRotationAxis(): set the rotation axis policy Static, RotationX or RotationY (default is RotationY)
- setRotationAngle(): set a peak rotation angle in deg (default is 0)
- setRotationSpeed(): set a peak rotation speed ixin deg/frame (default is 0)
- setGrowFactor(): set a growing factor (default is 1.0)
- setDiffractionPos(): set the source diplacement position x and y (default is center)
- setDiffractionSpeed(): set the source diplacement speed sx and sy (default is 0,0)

Std capabilites

This plugin has been implement in respect of the mandatory capabilites but with some limitations according to some programmer's choices. We only provide here extra information for a better understanding of the capabilities for the simulator camera.

- HwDetInfo

  The default (and max.) frame size if about 1024x1024-Bpp32, but one can only change the image type by calling DetInfoCtrlObj::setCurrImageType().

- HwSync

  Only IntTrig trigger mode is supported !! For both exposure time and latency time min. is 10e-9 and max. is 10e6.

Optional capabilites

In addition to the standard capabilities, we make the choice to implement some optional capabilities in order to have an improved simulation.

- HwShutter The simulator only support ShutterAutoFrame and ShutterManual modes.
- HwRoi

  There is no restriction for the ROI.

- HwBin

  Bin 1x1 or 2x2 only.

Configuration

No hardware configuration of course !!!

How to use

The LimaCCDs tango server provides a complete interface to the simulator plugin so feel free to test.

For a quick test one can use python, is this a short code example:

```
from Lima import Simulator
from lima impor Core
import time

cam = Simulator.Camera()
fb = Cam.getFrameBuilder()
# add a peak
p1 = Simulator.GaussPeak(10,10,23, 1000) # peak at 10,10 fwhm=23 and max=1000
fb.setPeaks([p1])
hwint = Simulator.Interface(cam)
control = Core.CtControl(hwint)

acq = control.acquisition()

# setting new file parameters and autosaving mode
saving=control.saving()

pars=saving.getParameters()
pars.directory='/tmp/'
pars.prefix='testsimul_'
pars.suffix='.edf'
pars.fileFormat=Core.CtSaving.EDF
pars.savingMode=Core.CtSaving.AutoFrame
saving.setParameters(pars)


# now ask for 2 sec. exposure and 10 frames
acq.setAcqExpoTime(2)
acq.setNbImages(10)

acq.prepareAcq()
acq.startAcq()

# wait for last image (#9) ready
lastimg = control.getStatus().ImageCounters.LastImageReady
while lastimg !=9:
  time.sleep(1)
  lastimg = control.getStatus().ImageCounters.LastImageReady

# read a image
im0 = control.ReadImage(0)
```

CHAPTER

FOUR

FUTURE CAMERAS

We have many contributors now for new camera plugins, main but not the least are: ESRF, SOLEIL, DESY, ALBA, FRMII, ANKA.

During the coming year, several new detector plugins will be available.

4.1 Under development

- Imxpad (ImXPAD)
- Pixirad (INFN)
- SVCam HR (SVS-Vistek)
- Vieworks

4.2 Foreseen

- Aviex + Espia board (ESRF)
- Quantum (linear detector)
- Mythen 2 (Dectris)
- Eiger (PSI)
- Sarnoff and Dalsa + Espia board (ESRF)

CHAPTER

FIVE

APPLICATIONS

5.1 Python TANGO server

This is the python Tango devices server by the ESRF team.

This server provides a main device for the standard camera control, a camera specific device for the camera configuration and a set of "plugin" devices for extra operations or just to export some subset of interface.

Thanks to the Lima framework, the control can be achieved through a common device as well as some features like calculations. The configuration of the detector is done by the specific detector device. At ESRF we decided to develop the Tango devices only in python language which implies that all the detector C++ interfaces have been wrapped to python modules.

5.1.1 Main device: LimaCCDs

LimaCCDs is the generic device and it provides a unique interface to control any of the supported cameras. One can find below the commands, the attributes and the properties.

To run a LimaCCDs server you will need at least to configure the **LimaCameraType** property. This property will tell to the LimaCCDs server which camera you want to run, then it will try to create the corresponding camera device (see the camera device chapter below for detailed list).

Property

Property name	Mandatory	Default value	Description
LimaCameraType	Yes	N/A	The camera type: e.g. Maxipix
NbProcessingThread	No	1	The max number of thread for processing. Can be used to improve the performance when more than 1 task (plugin device) is activated
AccThresholdCallbackModule	No	""	Plugin file name which manages threshold
ConfigurationFilePath	No	~/lima_<servname>.cfg	The default configuration file path
ConfigurationDefaultName	No	"default"	Your default configuration name

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrStringValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name
prepareAcq	DevVoid	DevVoid	Prepare the camera for a new acquisition, has to be called each time a parameter is set.
startAcq	DevVoid	DevVoid	Start the acquisition
stopAcq	DevVoid	DevVoid	Stop the acquisition
setImageHeader	DevVarStringArray: Array of string header	DevVoid	**Set the image header:** • [0]=”ImageId0 delimiter imageHeader0, • [1] = ImageId1 delimiter imageHeader1..
getImage	DevLong: Image number(0-N)	DevVarCharArray: Image data	Return the image data in raw format (char array)
getBaseImage	DevLong: Image number(0-N)	DevVarCharArray: Image data	Return the base image data in raw format (char array). Base image is the raw image before processing
readImage	DevLong: Image number(0-N)	DevEncoded: Encoded image	Return the image in encoded format of type “**DATA_ARRAY**” (see *data_array_encoded*)
writeImage	DevLong: Image number(0-N)	DevVoid	Save manually an image
readAccSaturatedImageCounter	DevLong: Image number(0-N)	DevVarUShortArray: Image counter	The image counter
readAccSaturatedImageCounter	DevLong: Image number	DevVarUShortArray: Image counter	The image counter
readAccSaturatedSumCounter	DevLong: from image id	DevVarLongArray: result	number of result for each images,sum counter of raw image #0 of image #0,sum counter of raw image #1 of image #0,...
setAccSaturatedMask	DevString	DevVoid	Full path of mask file, use empty string (“”) to unset the mask
closeShutterManual	DevVoid	DevVoid	Only if the camera has this capability
openShutterManual	DevVoid	DevVoid	Only if the camera has this capability
reset	DevVoid	DevVoid	Reset the camera to factory setting
getPluginNameFromType	DevString	DevString	Return the device name corresponding to the passed plugin named (.e.g FlatField)
configStore	DevVarStringArray:config	DevVoid	Store (im memory) a

Attributes

You will here a long list of attributes, this reflects the richness of the LIMA library. We organized them in modules which correspond to specific functions. A function module is identified by an attribute name prefix (excepted for informationnal attributes), for instance the **Acquisition** module attributes are always named **acq_<attr-name>**. The available modules are :

- General Information
- Status (prefix *last_* and *ready_*)
- Acquisition (prefix *acq_* for most of them sorry)
- Accumulation (prefix *acc_*)
- Saving (prefix *saving_*)
- Image (prefix *image_*)
- Shutter (prefix *shutter_*)
- Debug (prefix *debug_*)
- Video (prefix *video_*)
- Shared Memory (prefix *shared_memory_*)
- Configuration (prefix *config_*)

Many attributes are of type DevString and they have a fixed list of possible values. you can get the list by calling the special command **getAttrStringValueList**. Because a camera cannot support some attribute values , the command getAttrStringValueList will give you the the value list for the camera. For instance the attribute *video_mode* supports up to 14 different video formats, but a camera can only supports few of them.

Attribute name	RW	Type	Description
		GENERAL INFORMATION	
lima_type	ro	DevString	LImA camera type: Maxipix,Pilatus,Frelon,Pco, Basler ...
camera_type	ro	DevString	like lima_type but in uppercase !!
camera_pixelsize	ro	DevDouble[x,y]	The camera pixel size in x and y dimension
camera_model	ro	DevString	Camera model return by the detector layer:.e.g. 5x1-TPX1
plugin_type_list	rw		
write_statistic	ro	DevDouble[]	Performance writing time for last images in second.
		STATUS	
last_base_image_ready	ro	DevLong	The last base (before treatment) ready
last_image_ready	ro	DevLong	The last acquired image number, ready for reading
last_image_saved	ro	DevLong	The last saved image number

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
last_image_acquired	ro	DevLong	The last acquired image number
last_counter_ready	ro	DevLong	Tell which image counter is last ready
ready_for_next_image	ro	DevBoolean	True after a camera readout, otherwise false. Can be used for fast synchronisation with trigger mode (internal or external).
ready_for_next_acq	ro	DevBoolean	True after end of acquisition, otherwise false.
		ACQUISITION	
acq_status	ro	DevString	Acquisition status: Ready, Running, Fault or Configuration
acq_mode	rw	DevString	**Acquisition mode:** • **Single**, default mode one frame per image • **Concatenation**, frames are concatenated in image • **Accumulation**, powerful mode to avoid saturation of the pixel, the exposure is shared by multiple frames, see acc_ attributes for more
acq_nb_frames	rw	DevLong	Number of frames to be acquired, Default is 1 frame

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
acq_trigger_mode	rw	DevString	**Trigger mode:** • **Internal_trigger**, the software trigger, start the acquisition immediately after an acqStart() call, all the acq_nb_frames are acquired in an sequence. • **External_trigger**, wait for an external trigger signal to start the an acquisition for the acq_nb_frames number of frames. • **External_trigger_multi**, as the previous mode except that each frames need a new trigger input (e.g. for 4 frames 4 pulses are waiting for) • **Internal_trigger_multi**, as for internal_trigger except that for each frame the acqStart() has to called once. • **External_gate**, wait for a gate signal for each frame, the gate period is the exposure time. • **External_start_stop**

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
latency_time	rw	DevDouble	Latency time in second between two frame acquisitions, can not be zero, the minimum time corresponds to the readout time of the detector.
valid_ranges	ro	DevDouble[4]	min exposure, max exposure, min latency, max latency
concat_nb_frames	rw	DevLong	The nb of frames to concatenate in one image
acq_expo_time	rw	DevDouble	The exposure time of the image, Default is 1 second
		ACCUMULATION	
acc_expotime	ro	DevDouble	The effective accumulation total exposure time.
acc_nb_frames	ro	DevLong	The calculated accumulation number of frames per image.
acc_max_expotime	rw	DevDouble	The maximum exposure time per frame for accumulation
acc_time_mode	rw	DevString	**Accumulation time mode:** • **Live**,acq_expo_time = acc_live_time • **Real**,acq_expo_time = acc_dead_time + acc_live_time
acc_dead_time	ro	DevDouble	Total accumulation dead time
acc_live_time	ro	DevDouble	Total accumulation live time which corresponds to the detector total counting time.
acc_saturated_active	rw	DevBoolean	To activate the saturation counters (i.e. readAccSaturated commands)
acc_saturated_threshold	rw	DevLong	To fix the threshold for saturation counting

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
acc_saturated_cblevel	rw	DevLong	Set at which level in the process task chain this task will be called (default is 0,ie before other calculation task(mask,bg,flatfield ...)
		SAVING	
saving_mode	rw	DevString	**Saving mode:** • **Manual**, no automatic saving, a command will be implemented in a next release to be able to save an acquired image. • **Auto_Frame**, Frames are automatically saved according the saving parameters (see below). • **Auto_header**, Frames are only saved when the setImageHeader() is called in order to set header information with image data.
saving_directory	rw	DevString	The directory where to save the image files
saving_prefix	rw	DevString	The image file prefix
saving_suffix	rw	DevString	The image file suffix
saving_next_number	rw	DevLong	The image next number The full image file name is: • /saving_directory/saving_prefix+sp

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
saving_format	rw	DevString	**The data format for saving:** • **Raw**, save in binary format • **Edf**, save in ESRF Data Format • **edfgz** (or edf.gz), EDF with gz compression • **Tiff**, The famous TIFF format • **Cbf**, save in CBF format (a compressed format for crystallography)
saving_overwrite_policy	rw	DevString	**In case of existing files an overwite poli** • **Abort**, if the file exists the saving is aborted • **Overwrite**, if the file exists it is overwritten • **Append**, if the file exists the image is append to the file
saving_frame_per_file	rw	DevLong	Number of frames saved in each file
saving_common_header	rw	DevString[]	Common header with multiple entries
saving_header_delimiter	rw	DevString[]	The header delimiters, [0] = key header delimiter, [1] = entry header delimiter, [2] = image number header delimiter. Default : [0] = "=", [1] = "n", [2] = ";"
		IMAGE	

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
image_type	ro	DevString	**Return the current image data type, bi** • Bpp8, Bpp8S, Bpp10, Bpp10S, Bpp12, Bpp12S, Bpp14, • Bpp14S, Bpp16, Bpp16S, Bpp32, Bpp32S, Bpp32F.
image_width	ro	DevLong	Width size of the detector in pixel
image_height	ro	DevLong	Height size of the detector in pixel
image_sizes	ro	DevULong[4]	Signed(0-unsigned,1-signed), depth(nb bytes), width and height
image_roi	rw	DevLong[4]	Region Of Interest on image, [0] = Begin X, [1] = End X, [2] Begin Y, [3] = End Y, default ROI is [0,0,0,0] (no ROI)
image_bin	rw	DevLong[2]	Binning on image, [0] = Binning factor on X, [1] = Binning factor on Y. Default binning is 1 x 1
image_flip	rw	DevBoolean[2]	Flip on the image, [0] = flip over X axis, [1] flip over Y axis. Default flip is False x False
image_rotation	rw	DevString	Rotate the image: “0”, “90”, “180” or “270”
		SHUTTER	

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
shutter_mode	rw	DevString	Using the external shutter output to synchronize a shutter with the acquisition several modes are available: • **Manual** • **Auto_frame**, the output signal is activated for each individual frame of a sequence • **Auto_sequence**, the output signal is activated during the whole sequence
shutter_open_time	rw	DevDouble	Delay (sec.) between the output shutter trigger and the beginning of the acquisition, if not null the shutter signal is set on before the acquisition is started.
shutter_close_time	rw	DevDouble	Delay (sec.) between the shutter trigger and the end of the acquisition, if not null the shutter signal is set on before the end of the acquisition.
shutter_manual_state	rw	DevString	To open/close manually the shutter (if Manual mode is supported, see shutter_mode)
		DEBUG	
debug_module_possible	ro	DevString[]	Return the list of possible debug modules

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
debug_modules	rw	DevString[]	**Set the debug module level of LImA:** • "None" • "Common" • "Hardware" • "HardwareSerial" • "Control" • "Espia" • "EspiaSerial" • "Focla" • "Camera" • "CameraCom" • "Test" • "Application"
debug_types_possible	ro	DevString[]	Return the list of the possible debug types
debug_types	rw	DevString[]	**Set the debug type level of LImA:** • "Fatal" • "Error" • "Warning" • "Trace" • "Funct" • "Param" • "Return" • "Always"
		VIDEO	
video_active	rw	DevBoolean	Start the video mode (or not)
video_live	rw	DevBoolean	Start the video streaming (or not)
video_exposure	rw	DevDouble	The video exposure time (can be different to the acq_expo_time)
video_gain	rw	DevDouble	The video gain (if supported by the hardware)

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
video_mode	rw	DevString	**The video mode is the video format sup** • Y8, grey image 8bits • Y16, grey image 16bits • Y32, grey image 32bits • RGB555, color image RGB 555 encoding • RGB564, color image RGB 555 encoding • RGB24, color image RGB 24bits encoding • RGB32, color image RGB 32bits encoding • BGR24, color image BGR 24bits encoding • BGR32, color image BGR 32bits encoding • BAYER_RG8, color image BAYER RG 8bits encoding • BAYER_RG16, color image BAYER RG 16bits encoding • I420, color image I420 (or YUV420) planar encoding • YUV411, color image YUV411 planar encoding • YUV422, color image YUV422 planar encoding • YUV444, color image YUV444 planar encoding Depending of your camera, the supported formats can be retrieve using the command **getAttrStringValueList**

Continued on next page

Table 5.1 – continued from previous page

Attribute name	RW	Type	Description
video_roi	rw	DevLong[4]	A ROI on the video image (independent of the image_roi attribute)
video_bin	rw	DevULong[2]	A Binning on the video image (independt of the image_bin attribute)
video_last_image	rw	DevEncoded	The last video image, in DevEncoded "**VIDEO_IMAGE**" format, and using the video_mode set, see the DevEncoded definition *video_image_encoded*
video_last_image_counter	rw	DevLong64	The image counter
		SHARED MEMORY	
shared_memory_names	rw	DevString[2]	Firstname and surname of the SPS typed shared memory (default is LimaCCDs,<camera_type>)
shared_memory_active	rw		Activate or not the shared memory. The shared memory is for image display
		CONFIG	
config_available_module	ro	DevString[]	List of possible config modules,
config_available_name	ro	DevString[]	List of existing config names

DevEncoded DATA_ARRAY

The DATA_ARRAY DevEncoded has been invented for special Tango client like SPEC. It is used by the **readImage** command. It can only embed raw data (no video data). The supported image format can be retrieve with the **image_type** attribute (Bpp8,Bpp8S, ..., Bpp16,..) This encoded format is very generic and it supports many different type of data from scalar to image stack (see DataArrayCategory enumerate C-type). The readImage command only supports *Image* data array category.

The DATA_ARRAY format is composed of a fixed header followed by the raw data. The header is a C-like structure, with **little-endian** byte order and no alignment:

```
# The DATA_ARRAY definition
struct {
    unsigned int       magic= 0x44544159; // magic key
    unsigned short     version;           // version, only 1 supported
    unsigned  short    header_size;       // size of the header
    DataArrayCategory  category;          // data array category, see DataArrayCategory enumerate
    DataArrayType      data_type;         // data type, see DataArrayType enumerate
    unsigned short     endianness;        // 0-little-endian, 1-big-endian
    unsigned short     nb_dim;            // number of dimension (0 to 7 max)e.g 2 for image
    unsigned short     dim[8];            // size for each dimension, e.g [width,height]
    unsigned int       dim_step[8];       // step size in pixel for each dimension, e.g [1,height]
```

```
} DATA_ARRAY_STRUCT;

enum DataArrayCategory {
    ScalarStack = 0;
    Spectrum;
    Image;
    SpectrumStack;
    ImageStack;
};

enum DataArrayType{
    DARRAY_UINT8 = 0;
    DARRAY_UINT16;
    DARRAY_UINT32;
    DARRAY_UINT64;
    DARRAY_INT8;
    DARRAY_INT16;
    DARRAY_INT32;
    DARRAY_INT64;
    DARRAY_FLOAT32;
    DARRAY_FLOAT64;
};
```

DevEncoded VIDEO_IMAGE

The VIDEO_IMAGE DevEncoded has been implemented for the **video_last_image** attribute to return the last image. It can embed any of the supported video format depending of the **video_mode** attribute value.

The VIDEO_IMAGE format is composed of a fixed header followed by the data. The header is a C-like structure, with **big-endian** byte order and no alignment:

```
struct {
    unsigned int     magic_number = 0x5644454f;
    unsigned short   version;        // only version 1 is supported
    unsigned short   image_mode;     // Y8,Y16,....
    long     long    frame_number;   // the frame number (counter)
    int              width;          // the frame width in pixel (horizontal size)
    int              height          // the frame height in pixel (vertical size)
    unsigned short   endianness;     // 0-little-endian, 1-big-endian
    unsigned short   header_size;    // this header size in byte
    unsigned short   padding[2];     // 4 bytes of padding (for alignment)
} VIDEO_IMAGE_STRUCT;
```

5.1.2 Camera devices

Each camera has a configuration device with its own property/attribute/command lists. The camera configuration device is supposed to give you access to the "private" parameters of the detector that LIMA does not need but you may want to set. For instance some detectors provides a temperature control with set-points and/or start/stop commands for a auxillary cooling system.

For more details about the camera device interface, please have a look on the following sections:

Andor Tango device

This is the reference documentation of the Andor Tango device.

you can also find some useful information about prerequisite/installation/configuration/compilation in the *Andor camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
camera_number	No	N/A	The camera number, default is 0
config_path	No	N/A	The configuration path, for linux default is /usr/local/etc/andor
cooler	No	Off	Start/stop the cooling system of the camera mode
fast_trigger	No	Off	Fast external trigger mode, see Andor documentation for usage
shutter_level	No	High	The shutter output level mode
temperature_sp	No	N/A	The temperature setpoint in Celsius
p_gain	No	-1	The preamplifier gain [0-N] (see detector spec.), -1 to set to maxgain
vs_speed	No	-1	The vertical shift speed, index of speed [0-N], -1 to set fasten
adc_speed	No	-1	The adc/Horiz. speed pair index [0-N], -1 to set max speed

Attributes

Attribute name	RW	Type	Description
cooler	rw	DevString	**Start/stop the cooling system of the cam** • **On**, the cooler is started • **Off**, the cooler is stopped
cooling_status	ro	DevString	The status of the cooling system, tell if the setpoint temperature is reached
fast_trigger	rw	DevString	**Fast external trigger mode, see Andor** • **On**, fast mode, the camera will not wait until the a keep clean cycle has been completed before accepting the next trigger • **Off**, slow mode
shutter_level	rw	DevString	**The shutter output level mode:** • **Low"**, output TTL low signal to open shutter • **High**, output TTL high signal to open shutter
temperature	ro	DevShort	The current sensor temperature in Celsius
temperature_sp	rw	DevShort	The temperature setpoint in Celsius
timing	ro	Spectrum	The exposure and latency times
p_gain	rw	DevShort	The preamplifier gain [0-N] (see detector spec.), -1 to set to maxgain
vs_speed	rw	DevShort	The vertical shift speed, index of speed [0-N], -1 to set fasten recommended speed
adc_speed	rw	DevShort	The adc/Horiz. speed pair index [0-N], -1 to set max speed

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Basler Tango device

This is the reference documentation of the Basler Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Basler camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
cam_ip_address	Yes	N/A	The camera's ip or hostname
inter_packet_delay	No	0	The inter packet delay
frame_transmission_delay	No	0	The frame transmission delay

Both inter_packet_delay and frame_tranmission_delay properties can be used to tune the GiGE performance, for more information on how to configure a GiGE Basler camera please refer to the Basler documentation.

Attributes

This camera device has not attribute.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Dexela Tango device

This is the reference documentation of the Dexela Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Dexela camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
database_path	Yes	DexelaConfig.cfg	The database path file, e.g C:DexelaConfig.cfg
sensor_format	Yes	sensor2923	The detector model

Attributes

Attribute name	RW	Type	Description
full_well_mode	ro	DevString	The well-mode, can be set to HIGH or LOW

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Frelon Tango device

This is the reference documentation of the Frelon Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Frelon camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
espia_dev_nb	No	0	The acquisition Espia board number

Attributes

Attribute name	RW	Type	Description
espia_dev_nb	ro	DevString	The Espia board number.
image_mode	rw	DevString	**The acquisition image mode:** • **Frame transfert** • **Full frame**
input_channel	rw	DevString	**The Inputs ADC channels:** • **1** • **2** • **3** • **4** • **1-2** • **3-4** • **1-3** • **2-4** • **1-2-3-4**
e2v_correction	rw	DevString	**Active/Desactive the corrstion for e2v c** • **On** • **Off**
roi_mode	rw	DevString	**The roi mode:** • **None** • **Slow** • **Fast** • **Kinetic**
roi_bin_offset	rw	DevLong	The roi offset in line
spb2_config	rw	DevString	The internal config for pixel rate, **precision** or **speed**. Depending on your camera model, the pixel rates are factory defined
seq_status	ro	DevLong	

Please refer to the *Frelon User's Guide* for more information about the above specfic configuration parameters.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name
execSerial-Command	DevString command	DevString command result	Send a command through the serial line
resetLink	DevVoid	DevVoid	reset the espia link

ImXPAD Tango device

This is the reference documentation of the ImXPAD Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *ImXPAD camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
camera_ip_address	Yes	N/A	IP address
port	No	3456	socket port number
model	No	XPAD_S70	detector model
usb_device_id	No	N/A	reserved, do not use
config_path	Yes	N/A	The configuration directory path (see loadConfig command)

Attributes

This camera device has no attribute.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name
loadConfig	DevString	DevVoid	the config file prefix, the property config_path is mandatory

Maxipix Tango device

This is the reference documentation of the Maxipix Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Maxipix camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
config_name	Yes	N/A	The configuration name
config_path	Yes	N/A	The configuration directory path where the files are available
espia_dev_nb	No	0	The acquisition Espia board number
reconstruction_active	No	True	Activate the reconstruction or not
fill_mode	No	Raw	**the chip-gap filling mode, Raw, Zero, Dispatch** or **Mean**.
gate_level	No	High_Rise	The Input gate level, **High_rise** or **Low_Fall**
gate_mode	No	Inactive	The gate mode, **Inactive** or **Active**
ready_level	No	High_Rise	The output ready level, **High_rise** or **Low_Fall**
ready_mode	No	Exposure	The output Ready mode, **Exposure** or **Exposure_Readout**
shutter_level	No	High_Rise	The output Shutter level, **High_rise** or **Low_Fall**
trigger_level	No	High_Rise	The output Trigger level, **High_rise** or **Low_Fall**

Attributes

Attribute name	RW	Type	Description
config_name	rw	DevString	the configuration name. If changed the detector is re-configured and reset.
config_path	rw	DevString	the configuration directory path where the files are available
energy_calibration	rw	Spectrum DevDouble	The energy calibration, [0] = threshold setpoint , [1] threshold step-size (keV)
energy_threshold	rw	DevDouble	The threshold in energy (keV)
threshold	rw	DevDouble	The detector threshold
threshold_noise	rw	Spectrum DevDouble	The threshold noise of each chip, [0] =chip0 thl, [0] = chip1 thl, ...
espia_dev_nb	rw	DevString	The Espia board number.
fill_mode	rw	DevString	**The chip-gap filling mode:** • **Raw**, the border pixel values are copied • **Zero**, border and gap pixel are set to zero • **Dispatch**, the border pixel values are interpolated over the full gap • **Mean**, the gap pixels are filled with the border pixels average value.
gate_level	rw	DevString	**The Input gate level:** • **High_rise** • **Low_Fall**
gate_mode	rw	DevString	**The gate mode:** • **Inactive** • **Active**
ready_mode	rw	DevString	**The output Ready mode:** • **Exposure** • **Exposure_Readout**
shutter_level	rw	DevString	**The output Shutter level** • **High_rise** • **Low_Fall**

Warning: we recommend to not change the DAC register values (dac_name and dac_value attributes) excepted if you well know what you are doing, if you have some troubles with the detector please contact the ESRF Detector Unit first.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Merlin Tango device

This is the reference documentation of the Merlin Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Merlin camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
HostName	Yes	none	The detector IP address
CmdPort	No	6431	The tcp command port
DataPort	No	6432	The tcp data port
ImageWidth	No	512	The number of detector pixels
ImageHeight	No	512	The number of detector rasters
Chips	No	4	The number of detector medipix3 chips
Simulate	No	0	Command simulation mode

Attributes

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
SoftTrigger	DevVoid	DevVoid	Perform soft trigger
Abort	DevVoid	DevVoid	Abort
THScan	DevVoid	DevVoid	Perform threshold scan
ResetHW	DevVoid	DevVoid	Reset

Mythen3 Tango device

This is the reference documentation of the Mythen3 Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Xspress3 camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
HostName	Yes		The Mythen detector socket server IP address
TcpPort	No	1031	The tcp communication port.
Simulate	No	0	Command simulation mode.

Attributes

Attribute name	RW	Type	Description
acqRunning	ro	DevBoolean	Is acquisition active
assemblyDate	ro	DevString	Assembly date of the Mythen system
badChannelInterpolation	rw	DevString	Enable/Disable Bad Channel Interpolation Mode (**ON/OFF**)
badChannels	ro	DevLong[1280*Nb]	Display state of each channel for each active module [Nb = nbModules]
commandID	ro	DevLong	Command identifier (increases by 1)
continuousTrigger	rw	DevString	Enable/Disable continuous trigger mode (**ON/OFF**)
cutoff	ro	DevLong	Count value before flatfield correction
delayBeforeFrame	rw	DevLong64	Time delay between trigger & start (100ns increments)
energy	rw	DevFloat[Nb]	X-ray Energy (4.09 < e keV < 40) [Nb = nbModules]
energyMax	ro	DevFloat	Maximum X-ray Energy keV
energyMin	ro	DevFloat	Minimum X-ray Energy keV
flatField	ro	DevLong[1280*Nb]	Flat field correction values
flatFieldCorrection	rw	DevString	Enable/Disable Flat Field Correction Mode (**ON/OFF**)
gateMode	rw	DevString	Enable/Disable gate mode (**ON/OFF**)
gates	rw	DevLong	Number of gates per frame
hwStatus	ro	DevString	The hardware status
inputSignalPolarity	rw	DevString	Input Signal Polarity (**RISING_EDGE/FALLING_EDGE**)
kthresh	ro	DevFloat[Nb]	Threshold Energy (4.0 < e keV < 20) [Nb = nbModules]
kthreshEnergy	w	DevFloat[2]	Threshold & Energy keV
kthreshMax	ro	DevFloat	Maximum Threshold Energy keV
kthreshMin	ro	DevFloat	Minimum Threshold Energy keV
maxNbModules	ro	DevLong	Maximum nos. of Mythen modules
module	rw	DevLong	Number of selected module (-1 = all)
nbits	rw	DevString	Number of bits to readout (**BPP24/BPP16/BPP8/BPP4**)
nbModules	rw	DevLong	Number of modules in the system
outputSignalPolarity	rw	DevString	Output Signal Polarity (**RISING_EDGE/FALLING_EDGE**)
predefinedSettings	w	DevString	Load predefined energy/kthresh settings (**Cu/Ag/Mo/Cr**)
rateCorrection	rw	DevString	Enable/Disable rate correction mode (**ON/OFF**)
sensorMaterial	ro	DevLong	The sensor material (0=silicon)
sensorThickness	ro	DevLong	The sensor thickness um
serialNumbers	ro	DevLong[Nb]	Serial nos. of Mythen modules [Nb = nbModules]
systemNum	ro	DevLong	The serial number of the Mythen
tau	rw	DevFloat[Nb]	Dead time constants for rate correction [Nb = nbModules]
testPattern	ro	DevLong[1280*Nb]	Read back a test pattern
triggered	rw	DevString	Enable/Disable triggered mode (**ON/OFF**)
useRawReadout	rw	DevString	Raw readout packed Mode (**ON/OFF**)

Continued on next page

Table 5.2 – continued from previous page

Attribute name	RW	Type	Description
version	ro	DevString	The software version of the socket server

Commands

PCO Tango device

This is the reference documentation of the PCO Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *PCO camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
debug_control	No	0	To set a debug control level (hex value)
debug_module	No	0	To set the debug module list (hex value)
debug_format	No	0	To set the debug format (hex value)
debug_type	No	0	To set the debug type (hex value)

For the debug flags you can have a look on the **common/include/Debug.h** file.

Attributes

Attribute name	RW	Type	Description
rollingShutter	rw	DevString	**Set the rolling shutter mode:** • **'only for EDGE'** • **'GLOBAL'** • **'ROLLING'**
lastError	ro	DevString	The last PCO error message
camInfo	ro	DevString	read the *camInfo* takl parameter
camType	ro	DevString	read the *camType* talk parameter
clXferPar	ro	DevString	Read the *clTransferParam* talk parameter
cocRunTime	ro	DevString	Read the *cocRunTime* talk parameter
frameRate	ro	DevString	Read the *frameRate* talk parameter
pixelRate	ro	DevString	Read the *pixelRate* talk parameter
maxNbImages	ro	DevString	The maximum number of images which can be acquired by the camera
info	ro	DevString	Some information (again)
version	ro	DevString	Read the *timestamp* talk parameter

For the above attributes which use "talk" command you can refer to the PCO documentation for more information.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrStringValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name
talk	DevString	DevString	**WARNING**:use this command for test only, This is a backdoor cmd and it can distrub Lima

PerkinElmer Tango device

This is the reference documentation of the PerkinElmer Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *PerkinElmer camera plugin* section.

Properties

This device has no property.

Attributes

Attribute name	RW	Type	Description
correction_mode	rw	DevString	'NO', 'OFFSET ONLY' or 'OFFSET AND GAIN'
gain	rw	DevLong	The gain value, from 0 to 63
keep_first_image	rw	DevString	'YES' or 'NO', you can decide to trash the 1st image

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrStringValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name
startAcqOffsetImage	DevVarDoubleArray: nb_frames, exposure_time	DevVoid	Start acquisition for an offset calibration
startAcqGainImage	DevVarDoubleArray: nb_frames, exposure_time	DevVoid	Start an acquisition for an gain calibration

PhotonicScience Tango device

This is the reference documentation of the PhotonicScience Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *PhotonicScience camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
camera_library_path	Yes	N/A	the path to the camera DLL library file e.g.: ImageStar4022_v2.5imagestar4022control.dll

Attributes

This camera device has no attribute.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Pilatus Tango device

This is the reference documentation of the Pilatus Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Pilatus camera plugin* section.

Properties

This camera device has no property.

Property name	Mandatory	Default value	Description
TmpfsSize	No	0	OBSOLETE

Attributes

Attribute name	RW	Type	Description
threshold_gain	rw	DevString	The detector threshold gain (**LOW,MID,HIGH,ULTRA HIGH**)
fill_mode	rw	DevString	The gap fill mode (**ON,OFF**)
threshold	rw	DevLong	The threshold level of detector in eV
energy_threshold	rw	DevFloat	The energy threshold in keV (set the gain and the threshold)
trigger_delay	rw	DevDouble	The start exposure delay after the hard trigger
nb_exposure_per_frame	rw	DevLong	The number of exposure/frame to set an accumulation of frames Very useful to not saturate the pixel counters.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

PointGrey Tango device

This is the reference documentation of the PointGrey Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *PointGrey camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
camera_serial	Yes	N/A	The serial number of the camera, used to get the connection
packet_size	No	-1	The packet size, in byte
packet_delay	No	-1	The packet inter delay , in us last both parameters can be used to tune the camera GigE bandwidth, please refer to the camera documentation for more information

Attributes

Attribute name	RW	Type	Description
gain	rw	DevDouble	The camera gain factor, in dB
auto_gain	rw	DevBoolean	Auto gain mode can be switched on or off
auto_exp_time	rw	DevBoolean	The camera can be set to auto-exposure mode
auto_frame_mode	rw	DevBoolean	The camera can be set to auto frame rate mode
frame_rate	rw	DevDouble	The frame rate, in fps
packet_size	rw	DevLong	See the corresponding property
packet_delay	rw	DevLong	See the corresponding property
exp_time_range	ro	DevDouble[]	Return the exposure time range (min,max) in ms
gain_range	ro	DevDouble[]	Return the gain range (min,max) in dB
frame_rate_range	ro	DevDouble[]	Return the frame rate range (min,max) in fps

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrStringValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Prosilica Tango device

This is the reference documentation of the Prosilica Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Prosilica camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
cam_ip_address	Yes	N/A	The camera's ip or hostname

Attributes

This device has no attribute.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

RayonixHs Tango device

This is the reference documentation of the RayonixHs Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *RayonixHs camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
frame_mode	No	single	The frame mode, **single** or **fast_transfer**
frame_trigger_signal_type	No	opto	The frame trigger signal type (input #1)
sequence_gate_signal_type	No	opto	The gate signal type (input #2)
electronic_shutter_enabled	No	false	The electronic shutter **true** or **false** to activate or not
cooler_temperature_setpoint	No	-120	The cooling system temperature setpoint in Celsuis
sensor_temperature_setpoint	No	-80	The detector (sensor) temperature setpoint in Celsuis
output1_signal_type	No	cmos	The output #1 signal type
output2_signal_type	No	cmos	The output #2 signal type
output1_id	No	shutter	The output #1 signal source
output2_id	No	frame	the output #2 signal source

The Rayonix HS input/output system supports different type of signals:

- OPTO/OPTO_INVERTED/CMOS/CMOS_PULLDOWN/CMOS_PULLUP/CMOS_PULLDOWN_INVERTED/CMOS_P

And it provides a output multiplexer for both outputs within the following list of sources:

- SHUTTER/INTEGRATE/FRAME/LINE/SHUTTER_OPENING/SHUTTER_CLOSING/SHUTTER_ACTIVE/TRIGGER_

Attributes

Attribute name	RW	Type	Description
frame_mode	rw	DevString	The frame mode, **single** or **fast_transfer**
frame_trigger_signal_type	rw	DevString	The frame trigger signal type (input #1)
sequence_gate_signal_type	rw	DevString	The gate signal type (input #2)
electronic_shutter_enabled	rw	DevString	The electronic shutter **true** or **false** to activate or not
cooler_temperature_setpoint	rw	DevDouble	The cooling system temperature setpoint in Celsuis
sensor_temperature_setpoint	rw	DevDouble	The detector (sensor) temperature setpoint in Celsuis
output1_signal_type	rw	DevString	The output #1 signal type
output2_signal_type	rw	DevString	The output #2 signal type
output1_id	rw	DevString	The output #1 signal source
output2_id	rw	DevString	The output #2 signal source
vacuum_valve	rw	DevString	The vacuum valve command **true** or **false** to open or close

Warning: be careful with the temperature setting (and vacuum valve), the operating temperature is factory-determined and should never be changed. There is no reason to run the detector at a warner temperature.

For the signal type and source the possible values are listed above in the *Properties* section.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

RoperScientific Tango device

This is the reference documentation of the RoperScientific Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *RoperScientific camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
cam_num	Yes	N/A	The camera number

Attributes

Attribute name	RW	Type	Description
Temperature	rw	DevFloat	The detector read temperature
TemperatureSetPoint	rw	DevFloat	The detector temperature setpoint
Gain	rw	DevLong	The detector gain factor
InternalAcqMode	rw	DevString	The internal acquisition mode, **STANDARD**, **CONTINUOUS** or **FOCUS**
UseFullFrame	rw	DevLong	

Please refer to the PVCAM (SDK) user's guide for more information on how to use InternalAcqMode and UseFullFrame attributes.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrStringValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Simulator Tango device

This is the reference documentation of the Simulator Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Simulator camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
peaks	No	N/A	A gauss peak list [x0,y0,w0,A0,x1,y1,w1,A1...]
peak_angles	No	N/A	The base rotation angle for each peak
fill_type	No	Gauss	The image fill type: Gauss or Diffraction
rotation_axis	No	rotationy	Peak move policy: STATIC, ROTATIONX, ROTATIONY

Attributes

Attribute name	RW	Type	Description
peaks	rw	Spectrum,DevDouble	The gauss peak list [x0,y0,w0,A0,x1,y1,w1,A1...]
peak_angles	rw	Spectrum,DevDouble	The base rotation angle for each peak
grow_factor	rw	DevDouble	The Grow factor for gauss peaks
fill_type	rw	DevString	The image fill type: Gauss or Diffraction
rotation_axis	rw	DevString	The rotation axis policy: Static, RotationX or RotationY
diffraction_pos	rw	Spectrum,DevDouble	The source diplacement position: x and y
diffraction_speed	rw	Spectrum,DevDouble	The source diplacement speed: sx and sy
rotation_angle	rw	DevDouble	The peak rotation angle in deg
rotation_speed	rw	DevDouble	The peak rotation speed in deg/frame

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Ueye Tango device

This is the reference documentation of the Simulator Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Ueye camera plugin* section.

Properties

This device has no property neither attribute.

Attributes

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Ultra Tango device

This is the reference documentation of the Ultra Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Ultra camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
headIpaddress	No	192.168.1.100	The detector head IP address
hostIpaddress	No	192.168.1.103	The host IP address
tcpPort	No	7	The tcp echo port
udpPort	No	5005	The upd port
nPixels	No	512	The number of detector pixels

Attributes

Attribute name	RW	Type	Description
headColdTemp	ro	DevFloat	The head cold temperature in K
heatHotTemp	ro	DevFloat	The head hot temperature in K
tecColdTemp	ro	DevFloat	
tecSupplyVolts	ro	DevFloat	
adcPosSupplyVolts	ro	DevFloat	
adcNegSupplyVolts	ro	DevFloat	
vinPosSupplyVolts	ro	DevFloat	
vinNegSupplyVlots	ro	DevFloat	
headADCVdd	ro	DevFloat	
headVdd	rw	DevFloat	
headVref	rw	DevFloat	
headVrefc	rw	DevFloat	
headVpupref	rw	DevFloat	
headVclamp	rw	DevFloat	
headVres1	rw	DevFloat	
headVres2	rw	DevFloat	
headVTrip	rw	DevFloat	
fpgaXchipReg	rw	DevULong	
fpgaPwrReg	rw	DevULong	
fpgaSyncReg	rw	DevULong	
fpgaAdcReg	rw	DevULong	
frameCount	ro	DevULong	
frameError	ro	DevULong	
headPowerEnabled	rw	DevBoolean	
tecPowerEnabled	rw	Devboolean	
biasEnabled	rw	Devboolean	
syncEnabled	rw	Devboolean	
calibEnabled	rw	Devboolean	
8pCEnabled	ro	DevBoolean	
tecOverTemp	ro	DevBoolean	
adcOffset	rw	DevFloat[16]	
adcGain	rw	DevFloat[16]	
aux1	rw	DevULong[2]	
aux2	rw	DevULong[2]	
xchipTiming	rw	DevULong[9]	

Please refer to the manufacturer's documentation for more information about the above listed parameters and how to use them.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name
SaveConfigura-tion	DevVoid	DevVoid	Save the current configuration
RestoreConfig-uration	DevVoid	DevVoid	Restore the latest configuration

Xh Tango device

This is the reference documentation of the Xh Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Xh camera plugin* section.

Properties

Property name	Mandatory	Default value	Description
cam_ip_address	Yes	N/A	The detector IP address
port	No	1972	The port number
config_name	No	"config"	The default configuration filename

Attributes

Attribute name	RW	Type	Description
clockmode	wo	DevString	The clockmode, **XhInternalClock**, **XhESRF5468Mhz** or **XhESRF1136Mhz**
nbscans	wr	DevLong	the number of scans for accumulation

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name
reset	DevVoid	DevVoid	Perform a hardware reset of the detector
setHeadCaps	DevVarULongAr-ray	DevVoid	Caps for AB, Caps for CD
sendCommand	DevString	DevVoid	Backdoor command to send direct command to the *da.server* server

Xpad Tango device

This is the reference documentation of the Xpad Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Xpad camera plugin* section.

Properties

None.

Attributes

None.

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Xspress3 Tango device

This is the reference documentation of the Xspress3 Tango device.

you can also find some useful information about the camera models/prerequisite/installation/configuration/compilation in the *Xspress3 camera plugin* section.

test reference to camera plugin section: *Adsc camera plugin*

Properties

Property name	Mandatory	Default value	Description
basIpaddress	No	none	Override the base IP address (e.g. 192.168.0.1) from which all other addresses are calculated or NULL to use the default
basMacAddress	No	none	Override the base MAC address (e.g. 02.00.00.00.00) from which all other card MAC address‘s are calculated or NULL to use the default
basePort	No	none	Override the base IP port number or 0 to use the default
createScopeModule	No	False	true = do not create a scope data module
nbFrames	No	1	Number of 4096 energy bin spectra timeframes
scopeModName	No	NULL	The scope data module filename or NULL to use the default
nbCards	No	1	The number of xspress3 cards that constitute the xspress3 system, between 1 and XSP3_MAX_CARDS
nbChans	No	-1	Limit the number of channels
debug	No	0	debug message (0 = off, 1=normal, 2=verbose)
noUDP	No	False	True = do not do UDP connection
cardIndex	No	none	Starting card index
directoryName	No	non	The directory name to save and restore configurations

Attributes

Attribute name	RW	Type	Description
card	rw	DevLong	
numChan	ro	DevLong	
numCards	ro	DevLong	
chansPerCard	ro	DevLong	
maxNumChan	ro	DevLong	
binsPerMca	ro	DevLong	
windows	rw	DevLong[32]	
runMode	rw	DevBoolean[4]	
clocks	rw	Devbooleanp[3]	
goodsThreshold	rw	DevLong[16]	
dtcEnergy	rw	DevDouble	
dtcParameters	rw	DevDouble[48]	
scaling	rw	DevDouble[8]	
fanTemperatures	rw	DevDouble[50]	
fanController	rw	DevDouble[2]	
setPoint	wo	DevDouble	
roi	wo	DevLong[25]	
useDtc	rw	DevBoolean	
setTiming	wo	DevLong	
adcTempLimit	wo	DevLong	
setPlayback	wo	DevBoolean	
playbackfilename	wo	DevString	
dataSource	rw	DevLong[8]	

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name
Reset	DevVoid	DevVoid	
InitBrams	DevLong: channel	DevVoid	
Pause	DevVoid	DevVoid	
Restart	DevVoid	DevVoid	
Arm	DevVoid	DevVoid	
Clear	DevVoid	DevVoid	
SaveSettings	DevVoid	DevVoid	
RestoreSettings	DevBoolean	DevVoid	Force restore if major revision of saved file does not match the firmware revision
InitRois	DevLong: channel	DevVoid	
ReadHistogram	DevVarLongArray: frame, channel	DevVarULongArray:	Return the histogram data
ReadScalers	DevVarLongArray: frame, channel	DevVarULongArray:	Return the scaler data
StartScope	DevVoid	DevVoid	
LoadPlayback	DevVarLongArray: src0,src1, [num_streams, digital]	DevVoid	
FormatRun	DevVarLongArray: chan,[nbits_eng, aux1_mode, adc_bits, min_samples, aux2_mode, pileup_reject	DevVoid	

5.1.3 Plugin devices: software operation and extra interfaces

User-defined software plugins can be used to execute arbitrary image-based operations. An entry point in the control layer completely exports the ProcessLib functionality, allowing an external code to be called on every frame. The software operation can be implemented in C++ or Python.

The software operations on image are embedded into individual Tango devices and are available in the **plugins/** directory. They are automatically exported by the LimaCCDs server.

The software operations are of two types, *Sink* or *Link* :

- **Link** operation is supposed to modify the frame data, so it gets the frame data as input parameter and it will return a "corrected" image (e.g. Mask/Flatfield/BackgroundSubstraction).
- **Sink** operation is taken the frame data as input parameter to apply some software operation in order to return new data like statistics, peak positions, alarm on saturation ... etc.

In addition to sink/link plugin device, a plugin can just be implemented to provide/export a subset of the Lima interface or a legacy interface for some specific client applications (e.g SPEC, LimaTacoCCD plugin).

Today there are about 8 standard plugin devices:

- BackgroundSubstraction : link operation, to correct the frames with a background image (substraction)
- FlatField: link operation to correct the frames with a flatfield image (divide + option normalisation)
- Mask: link operation to mask pixels. Very useful if some pixel are not working properly and if you want to set then to a fix value or to zero.
- PeakFinder: thanks to Teresa Numez from DESY, a sink operation which can detect diffraction peaks.
- Roi2Spectrum: sink operation to apply ROI spectrum on the frames. You can define more than one spectra with ROI coordinates and by specifying in which direction you need to bin the values, vertical or horizontal.
- RoiCounter: sink operation to get calculating statistics on image regions.
- LimaTacoCCD: extra interface for TACO clients, it only provides commands (TACO does not have attribute !), it is still used at ESRF for SPEC.
- LiveViewer: extra interface to provide a live view of the last acquired image, can be used from atkpanel.

If you need to implement your own plugin device we can provide you some example codes, use the mailing-list lima@esrf.fr to get help.

Background Substraction

The Background substraction correction is a simple operation you can active when a detector has some dark-current noise independent of the dose of photons it will receive. To set the correction you must provide to the device a background image (**setBackgroundImage** cmd) file and then start (**start** cmd) the correction.

Properties

This device has no property.

Attributes

Attribute name	RW	Type	Description
RunLevel	rw	DevLong	Run level in the processing chain, from 0 to N
delete_dark_after_read	rw	DevBoolean	If true the device will delete the file after reading Can be useful to not keep obsolete dark image file after use

Commands

Command name	Arg. in	Arg. out	Description
setBackgroundImage	DevString	DevVoid	Full path of background image file
Start	DevVoid	DevVoid	Start the correction for next image
Stop	DevVoid	DevVoid	Stop the correction after the next image

FlatField

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Attributes

Attribute name	RW	Type	Description
TODO	ro	DevShort	todo

Properties

Property name	Mandatory	Default value	Description
TODO	No	N/A	Todo

Mask

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Attributes

Attribute name	RW	Type	Description
TODO	ro	DevShort	todo

Properties

Property name	Mandatory	Default value	Description
TODO	No	N/A	Todo

PeakFinder

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Attributes

Attribute name	RW	Type	Description
TODO	ro	DevShort	Todo

Properties

Property name	Mandatory	Default value	Description
TODO	No	N/A	Todo

Roi2Spectrum

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Attributes

Attribute name	RW	Type	Description
TODO	ro	DevShort	todo

Properties

Property name	Mandatory	Default value	Description
TODO	No	N/A	Todo

RoiCounter

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Attributes

Attribute name	RW	Type	Description
TODO	ro	DevShort	todo

Properties

Property name	Mandatory	Default value	Description
TODO	No	N/A	Todo

LimaTacoCCD

This device has been created by legacy and it provides the only interface that SPEC software is supporting for "ESRF General CCD Dev" CCD-like controller.

Properties

Property name	Mandatory	Default value	Description
ManualAsyn-chronousWrite	No	False	Flag for manual writting, can improve the performance of data saving

Attributes

This device has no attributes.

Commands

LiveViewer

Commands

Command name	Arg. in	Arg. out	Description
Init	DevVoid	DevVoid	Do not use
State	DevVoid	DevLong	Return the device state
Status	DevVoid	DevString	Return the device state as a string
getAttrString-ValueList	DevString: Attribute name	DevVarStringArray: String value list	Return the authorized string value list for a given attribute name

Attributes

Attribute name	RW	Type	Description
TODO	ro	DevShort	todo

Properties

Property name	Mandatory	Default value	Description
TODO	No	N/A	Todo

CHAPTER

SIX

CAMERA PLUGIN DEVELOPER GUIDE

6.1 Understand the plugin architecture

6.1.1 Library structure

The library structure is divided into two main layers: the control, containing the common control and processing code, and the hardwarewhich is implementing the detector-specific part. The control layer provides the library interface to the high level application. User requests to configure and control the acquisition are gathered by the control layer, so the hardware layer functionality is limited to the generation the image frames in a best-effort basis. The control layer is responsible of: i) adapting the received image geometry if it does not match the user requests, and ii) execute the frame processing chain.

6.1.2 Generic Interface

The Hardware Layer defines the interface between the Control Layer and the controller library. It provides the minimal functionality needed for the Control Layer to satisfy the user requests. The main class in the Hardware Layer is the HwInterface, providing the interface to the Control Layer. In order to provide a flexible and evolvable interface, the configuration of this layer is implemented as a set of features (capabilities) that may or may not be implemented by the hardware. The capabilities can be grouped in three categories:

1. Standard: includes the synchronization parameters (exposure time, ext. trigger, etc), the detector informations (Detector model, Max size, etc..) is considered standard and must be implemented for all detectors.
2. Extended*: optional common features like image transformations (binning, RoI, flip), advanced acquisition modes (kinetics, frame transfer), and extended mechanisms (camera serial line)
3. Specific*: these are detector-specific features that can not be treated in a generic interface

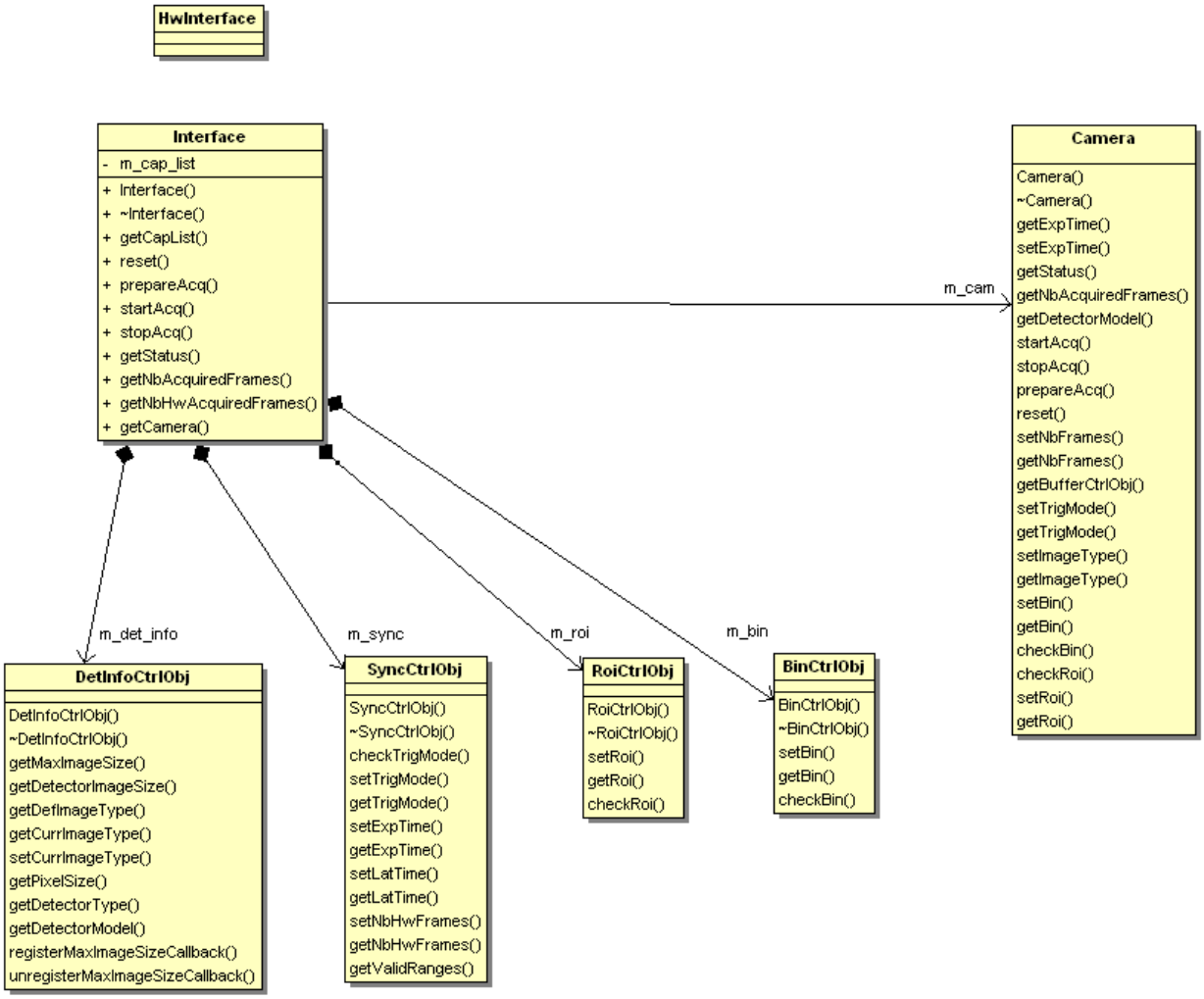
As a camera plugin developer your main job will consist in writting the code for the HwInterface class and its depending classes (.e.g the capabilities classes). The figure 1 illustrates the class relationships:

Figure 1.

6.1.3 Hardware Interface: HwInterface

As an interface to the Control Layer, this class exports the capabilities provided by the hardware: * getCapList() It also contains the high level commands to control the acquisition:

- prepare()
- start()


HwInterface
Interface
- m_cap_list
+ Interface()
+ ~Interface()
+ getCapList()
+ reset()
+ prepareAcq()
+ startAcq()
+ stopAcq()
+ getStatus()
+ getNbAcquiredFrames()
+ getNbHwAcquiredFrames()
+ getCamera()
m_cam
Camera
Camera()
~Camera()
getExpTime()
setExpTime()
getStatus()
getNbAcquiredFrames()
getDetectorModel()
startAcq()
stopAcq()
prepareAcq()
reset()
setNbFrames()
getNbFrames()
getBufferCtrlObj()
setTrigMode()
getTrigMode()
setImageType()
getImageType()
setBin()
getBin()
checkBin()
checkRoi()
setRoi()
getRoi()
m_det_info
m_sync
m_roi
m_bin
DetInfoCtrlObj
DetInfoCtrlObj()
~DetInfoCtrlObj()
getMaxImageSize()
getDetectorImageSize()
getDefImageType()
getCurrImageType()
setCurrImageType()
getPixelSize()
getDetectorType()
getDetectorModel()
registerMaxImageSizeCallback()
unregisterMaxImageSizeCallback()
SyncCtrlObj
SyncCtrlObj()
~SyncCtrlObj()
checkTrigMode()
setTrigMode()
getTrigMode()
setExpTime()
getExpTime()
setLatTime()
getLatTime()
setNbHwFrames()
getNbHwFrames()
getValidRanges()
RoiCtrlObj
RoiCtrlObj()
~RoiCtrlObj()
setRoi()
getRoi()
checkRoi()
BinCtrlObj
BinCtrlObj()
~BinCtrlObj()
setBin()
getBin()
checkBin()

- stop()
- getStatus()
- getNbAcquiredFrames()
- getStartTimeStamp()

The getStatus() function should return the following information:

Parameters	Description
AcqStatus	Global acquisition status: Ready/Running/Fault
DetStatus	**Compound bit flags specifying the current detector status:** • Fault • WaitForTrigger • ShutterOpen • Exposure • ShutterClose • ChargeShift • Readout
DetStatusMask	A mask specifying the detector status bits that are supported by the hardware

6.1.4 Standard Capabilities

These capabilities are mandatory for all the detectors. They define the minimum functionality necessary for image acquisition. Three capability classes (DetInfo, Sync and BuffCtrl) are listed below with their set/get methods which have to be provided within the new camera plugin code.

Detector Information: HwDetInfoCtrlObj

This capability returns static information about the detector and the current image dimension.

Parameters	Description
getMaxImageSize	Maximum size of the image
getDetectorImageSize	Size of the detector image, it is always equal or greater than the MaxImageSize
getDefImageType	Default data type of image (ushort, ulong, ...)
getCurrImageSize	Current maximum size of the image, excluding binning and RoI
getCurrImageType	Data type of image (ushort, ulong, ...)
getPixelSize	Physical size of pixels (in mm)
getDetectorType	Type of the detector (Frelon, Maxipix, ...)
getDetectorModel	Model of the detector

In addition to those parameters two more methods have to be implemented for a callback function:

- registerMaxImageSizeCallback
- unregisterMaxImageSizeCallback

This callback function is to inform the Lima library hardware layer of a change of the detector maximum image size.This change can happen with some detectors which can be reconfigured with a different geometry. This camera capability is **NOT** a Roi **nor** a Bin capability. For instance, the maxipix detector is a mosaic of several individual sensor chips and it can be configured and reconfigured with different geometries according to user needs. A 2x2 maxipix detector can be configured in a 1x1 geometry.

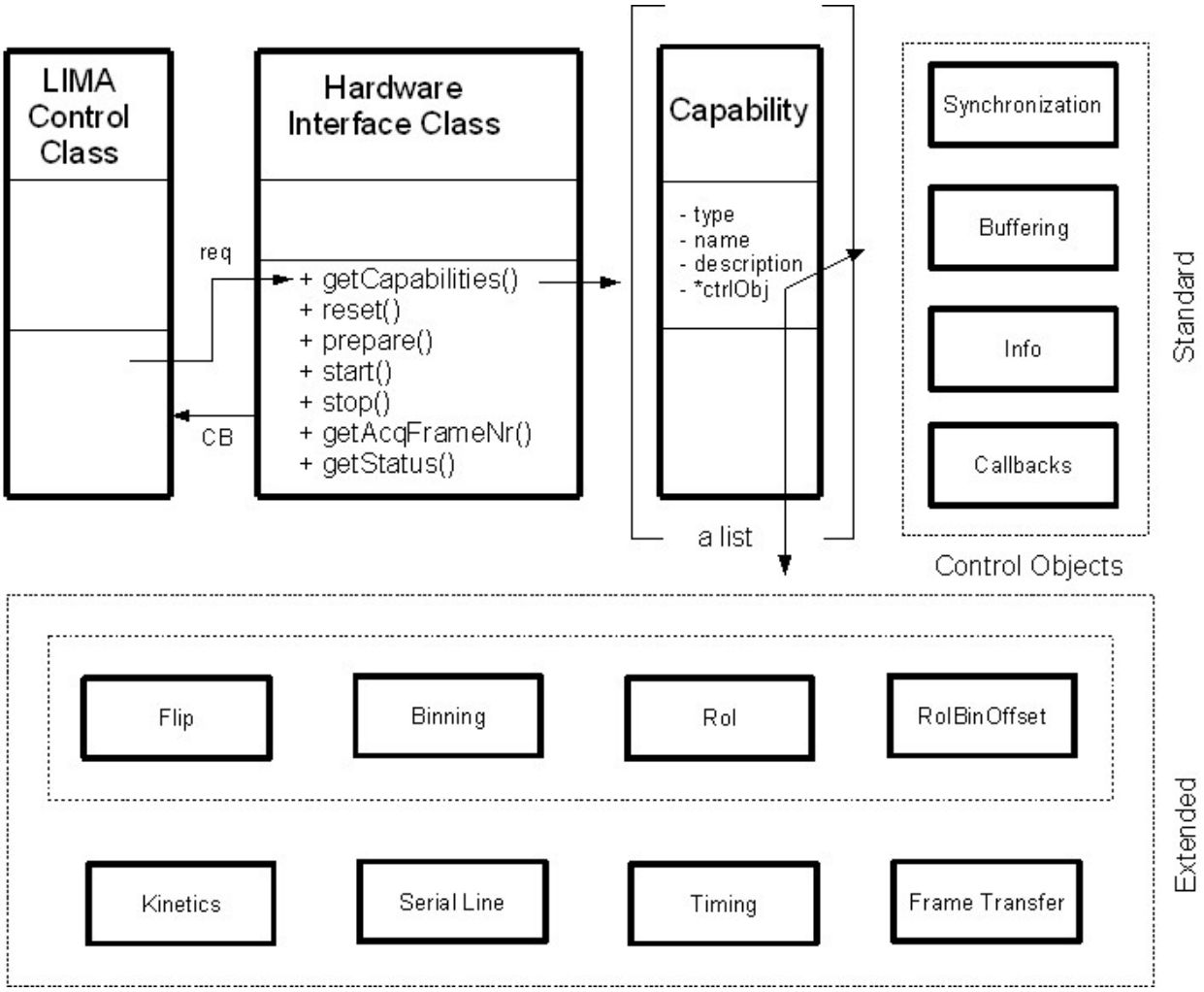


Figure 6.1: Figure 2.

Synchronization: HwSyncCtrlObj

This capability return/set acquisition parameters.

Parameters	Description
set/getExpTime	Frame exposure time
set/getLatTime	Latency time between frames
checkTrigMode	A check method which returns True/False for the supported trigger modes
set/getTrigMode	**Triggering mode:** • Internal: software triggering • ExtStart: one external signal to start the whole sequence acquisition (one or more frames per sequence) • MultExtStart: one external signal for each frame in the acquisition sequence • Gate: controls start and stop of each frame • ExtStartStop: one start signal to start acquisition of one frame and one signal to stop it

Buffer Management: HwBufferCtrlObj

This block controls the image memory buffer allocation and management. They are used: * As temporary frame storage before saving, allowing disk/network speed fluctuations. * To permanently hold images that can be read by the user after the acquisition is finished.

These buffer functionalities may be implemented by the hardware layer (kernel driver in the case of the Espia). If not, an auxiliary buffer manager class will be provided to facilitate (and unify) its software implementation. The buffer management parameters are:

Parameters	Description
NbBuffers	Number of image buffers in memory.
NbConcatFrames	The number of concatenated frames per buffer.
NbAccFrames	The number of detector frames to accumulate into a single buffer.
MaxNbBuffers	This Read-Only parameter indicates the maximum number of buffers that can be allocated, given the size of the frame and the number of (concatenated) frames per buffer.
BufferMode	Buffer filling mode (linear or circular)

The buffer manager must also provide the following methods: * getBufferPtr(int buffer_nb) * getFramePtr(int acq_frame_nb) * getFrameTimeStamp(int acq_frame_nb)

In most of simple cases, one just need to create a **SoftBufferCtrlObj** class instance within the Camera class instance to store the frames. A good example of a simple implementation is available in the Andor camera plugin code.

Frame callback

The hardware must provide callbacks after each acquired frame. The callback function should receive the following information:

Parameters	Description
AcqFrameNb	Index of the frame since the start of the acquisition
FramePtr	Pointer to the frame memory
FrameDim	Structure holding the width, height and type of the frame
TimeStamp	Time (in sec.) since the start of the acquisition

The frame callbacks are implemented by means of an auxiliary class (FrameCallback), which will be used by the Control Layer. From the Hardware Layer point of view, the standard capability control object must implement two functions:

- setFrameCallbackActive(bool cb_active)
- frameReady(<callback_frame_info>)

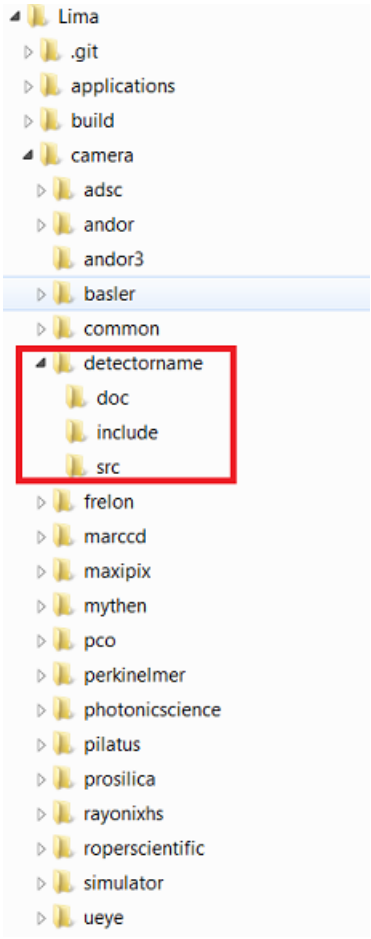
6.2 Organize the source code

This chapter provides general guidelines to follow, to share a plugin with the **LIMA** community.

6.2.1 Source code

- **Plug-ins directory**

  The source files and documentation of each new plug-in must be located under Lima/Camera as shown figure below.



To maintain homogeneity between the different plug-ins, each plug-in must have at minimum the following folders:

- **src** : contains the source files. Plug-ins must be developed in C++.

  The "src" folder must contain the following files :

  - DetectorNameCamera.cpp : interface class with the detector librairies **(mandatory)**
  - DetectorNameInterface.cpp : interface class between detector capabilities from the hardware interface and the control layer **(mandatory)**
  - DetectorNameDetInfoCtrObj.cpp : capabilities to get static informations about the detector **(mandatory)**
  - DetectorNameBufferCtrlObj.cpp : capabilities to control the image memory buffer allocation **(mandatory)**
  - DetectorNameSyncCtrlObj.cpp : capabilities to control the image memory buffer allocation **(mandatory)**
  - DetectorNameRoiCtrlObj.cpp : capabilities to get a ROI **(optional)**
  - DetectorNameBinCtrlObj.cpp : capabilities to make pixel binning **(optional)**
  - DetectorNameVideoCtrlObj.cpp : capabilities to make video mode only for non-scientific detectors **(optional)**

- DetectorNameShutterCtrlObj.cpp : capabilities to control shutter **(optional)**
- DetectorNameFlipCtrlObj.cpp : capabilities to flip image **(optional)**
- DetectorNameEventCtrlObj.cpp : capabilities to generate event **(optional)**
- DetectorNameSavingCtrlObj.cpp : capabilities to save images in different formats **(optional)**

If optional capabilities are not defined, they are emulated by the Lima Core.

- **include** : contains the header files relative to the sources files described before.
- **doc** : contains at least "index.rst" for plug-in documentation. Other files such as image can be added. The minimum content of the index file is detailed in the documentation section.
- **Other** : Other folders can be added based on need. The contents of this file must be described in the documentation.

- **Camera device**

  Once the plug-in was developed, you must create a camera device to execute all commands on the camera. This device can be developed in Python or C++. Python devices must be located on "Lima/applications/tango/camera", C++ devices on "Lima/applications/tango/LimaDetector"

  It is recommended that the camera device comply with the design guidelines of a TANGO device. These few rules are described below :

  - **Tango design guidelines**

    In order to enhance the general software quality of Device Servers developed by the various institutes using Tango, a Design and Implementation Guidelines document has been written by SOLEIL. This document can be downloaded form thr URL : 'http://www-controle.synchrotron-soleil.fr:8001/docs/TangoGuidelines/TangoDesignGuidelines-GB4-3.pdf '

6.2.2 Class names

Again, to maintain homogeneity, it is recommended to follow this nomenclature for the class names:

- **DetectorName**::Camera
- **DetectorName**::Interface
- **DetectorName**::SyncCtrlObj
- **DetectorName**::DetInfoCtrlObj

As an example, one can look at the Prosilica plugin for a real implementation or at the simulator plugin for a implementation model.

6.2.3 How to test the new plugin with python

In order to communicate with the underlying detector hardware, the lima client must instantiate the main object of the framework Lima: CtControl. To be instantiated, the CtControl requires an interface inherited from common HwInterface. This interface requires the Camera object that encapsulates dependency with detector and its API driver.

For instance if you are using the python binding for the Prosilica camera, a client application initialization should do:

```
from Lima import Prosilica as ProsilicaAcq
from Lima import Core

my_prosilica_ip_address = 192.168.1.2
# we need the camera object first
camera = Prosilica.camera(my_prosilica_ip_address)

# create the HwInterface which needs the camera as unique parameter
camera_interface =  ProsilicaAcq.Interface(camera)

# Now create the CtControl and passed to Lima the new HwInterface
Control = Core.CtControl(camera_interface)
```

The camera is now under control and it can be used to acquire images ! First get the sub-objects for the parameter setting of the detector, acquisition, saving and more if necessary.

```
acq = control.acquisition()
saving = control.saving()

acq.setAcqExpoTime(0.1)
acq.setAcqNbFrames(10)

pars=saving.getParameters()
pars.directory='/buffer/test_lima'
pars.prefix='test1_'
pars.suffix='.edf'
pars.fileFormat=Core.CtSaving.EDF
pars.savingMode=Core.CtSaving.AutoFrame
saving.setParameters(pars)

# pass parameters to camera hw interface
control.prepareAcq()

# start the acquisition
control.startAcq()
```

NB: Camera object is only used to enhance the separation between the generic interface and the API driver of the detector. It is similar to a proxy.

The camera class is also supposed to provide an access to the specific configuration of the detector. For instance if your detector has a threshold setting or a built-in background correction available you should implement these features in the Camera class. The HwInterface will not know about the specific configuration and a client application should explicitly implement the configuration. A good example is the Andor camera, where there are few extra features like the temperature set-point (set/getTemperatureST()) or the cooler control (set/getCooler(bool)).

With the Andor camera one can set the cooling as:

```
camera.setTemperatureSP(-50)
camera.setCooler(True)

current_temp = Camera.getTemperature()
```

The Lima project code provides some client application based on TANGO protocol for the remote access. One can find a python implementation under applications/tango and a C++ version in applications/tango/LimaDetector. The python server has been developed at ESRF and being used on lot of beamlines and the C++ server is the SOLEIL version which is also used on beamlines.

The LimaCCDs python server has its own documentation here: http://lima.blissgarden.org/applications/tango/doc/index.html
.

6.3 Do not forget to write a documentation

Plugin documentation must be located in “Lima/camera/dectector/name/doc”. It is composed of at least an “index.rst” file which contains informations to install, configure and implement a camera plugin. The presence of this documentation is required to share a plugin with Lima community.

Plugins documentation is available in the section “Supported Cameras”.

The table below describes informations taht must be present in the index file :

Detector Name

Picture of the detector

Introduction

In this section you should describe the detector :
- Manufacter, model
- Interface buses (USB, GIGE, CameraLink, specific acquisition boards,...)
- Type of applications (scientific, industrial, medical, ...)
- OS Supported

Prerequisite

In this section you should specify librairies, driver or software packages required to compile the plugin :
- Version
- Installation path
- Specific procedure for installation (script to execute, environment variables,...)

Installation & Module configuration

In this section you should describe specific procedure for plugin installation :
- Configuration file "config.inc"
- Post installation actions
- Refer to the installation section to compile and install the plugin

Capabilities

Standard capabilities :

Although the plugin as been implement in respect of the mandatory capabilities, some limitations which are due to the camera and SDK features can exist. You should provide here extra information for a better understanding of the three mandatory capabilities below :
- HwDetInfo
- HwSync
- HwBuffer

Optional capabilities :

If optional capabilities are supported by the detector, they should be listed in this section. If some limitiations exist, they should be described here. Available optional capabilities are :
- HwRoi
- HwBin
- HwVideo
- HwShutter
- HwFlip
- HwEvent
- HwSaving

Configuration

This section must summarize different actions to configure the device server and the camera :
- Procedure to configure camera (external tools to set ip adress, ...)
- Properties of the device server to configure
- How to connect the camera
- Others

How to use

In this section you should give a code example to test the plugin. Code may be written in C++ or Python

CHAPTER

SEVEN

CLASS DOCUMENTATION

CHAPTER

EIGHT

INDICES AND TABLES

- *genindex*
- *modindex*
- *search*